VISIT…



Alexander I. Galushkin

Neural Networks Theory

Springer

Alexander I. Galushkin

Neural Networks Theory

Alexander I. Galushkin

Neural Networks Theory

With 176 Figures

Springer

Author

Professor Alexander I. Galushkin

Moscow Institute of Physics & Technology
Department Neurocomputers
Institutskiy per 9
141700 Dolgoprudniy
Moskva Region
Russia
E-mail: neurocomputer@yandex.ru

Library of Congress Control Number: 2007931627

ISBN 978-3-540-48124-9 **Springer Berlin Heidelberg New York**

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitations, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media
springer.com
© Springer-Verlag Berlin Heidelberg 2007

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover design: Erich Kirchner, Heidelberg
Production: Almas Schimmel
Typesetting: Büro Stasch, Susanne Kirchhofer, Bayreuth (stasch@stasch.com)

Printed on acid-free paper 30/3180/as – 5 4 3 2 1 0

This book is dedicated to my teachers, Russian classics in the field of control theory and theory of discrete and adaptive systems:
Vladimir V Solodovnikov, Lev T. Kuzin, and Yakov Z. Tsypkin

Foreword

Professor A. I. Galushkin's monograph "Neural Networks Theory" appears at a time when the theory has achieved maturity and is the fulcrum of a vast literature. Nevertheless, Professor Galushkin's work has high importance because it serves a special purpose which is explained in the following.

The roots of neural networks theory go back to the pioneering work of McCulloch and Pitts in the early 1940s. As a graduate student at MIT and later as an instructor at Columbia University, I was a witness to the birth of the digital world that took place in the years following the end of World War II and the beginning of the Cold War. To me, the definitive event was Shannon's first lecture on information theory which took place in New York in 1946. I attended his lecture and was utterly fascinated by what I heard. Wiener's cybernetics and the invention of the transistor were the other defining events which marked the debut of information revolution and the era of machine intelligence. The pioneering work of McCulloch and Pitts was too embryonic to attract much attention.

The work of McCulloch and Pitts was followed in the early 1950s by the development of threshold logic for pattern classification and automata theory for systems analysis. These developments were the backdrop for Frank Rosenblatt's invention of the perceptron – the forerunner of multilayer neural networks. Frank Rosenblatt was a visionary who believed that the perceptron could perform miracles. Unfortunately, his lectures were hard to follow and not persuasive. His revolutionary ideas were not accorded recognition in his prematurely ended lifetime.

In the early 1960s at Moscow's famed Institute of Automatics and Telemechanics, Ya Z. Tsypkin, M. A. Aizerman and others began to develop a theory of adaptive systems which led to the initiation of research on neural networks in the Soviet Union and to later work of Vapnik and Cherevonkis on support vector machines and kernel methods. Professor Galushkin was a student of Ya Z. Tsypkin and has played a pivotal role in the development of neural networks theory and its applications in the Soviet Union ever since.

Development of neural networks theory in the Soviet Union paralleled and, in some areas, especially in the realm of back propagation, was ahead. A detailed comparison and an overview are presented in Chap. 17 of Professor Galushkin's work.

The extensive development of neural networks theory in the Soviet Union was largely unknown in the Western world. One of the objectives of Professor Galushkin's work is to bring this fact to light. In this perspective, Professor Galushkin's monograph serves an important purpose. But perhaps more importantly, his work stands out as an au-

thoritative, comprehensive and up-to-date account of neural networks theory and its wide-ranging applications. Particularly worthy of note are Professor Galushkin's expositions of optimal models of neural networks, structure optimization by topological characteristics, continual neural networks, optimal neural networks for multidimensional signals, multivariable function extremum search algorithms, random search algorithms for local and global extrema, implementation of minimum average risk function criteria, closed-loop neural networks operating on nonstationary patterns, selection of initial conditions for neural network adjustment, design of neural networks for matrix inversion problems, informative feature selection, multilayer neural network functional liability, and neural network diagnostics. In these expositions, there is much that is new and not readily found in the Western literature.

There is a significant issue which is not addressed in Professor Galushkin's work, namely, the role of neural networks theory in soft computing. In science, as in most other realms of human activity, there is a tendency to adopt a particular methodology and march under its banner, in the belief that this methodology is superior to all others. The principal thesis of soft computing is that there is much to be gained by forming an alliance of various computing methodologies and using them in combination rather than in a stand-alone mode. In soft computing, the principal members of the alliance are fuzzy logic, neurocomputing, evolutionary computing, probabilistic computing and machine learning, and the principal objective is to provide a foundation for the conception, design and utilization of intelligent systems. In recent years, a combination which has received a great deal of attention is that of neurofuzzy systems. In such systems, the capability of neural networks to deal with classification, identification and adaptation is combined with the capability of fuzzy systems to deal with imprecision, uncertainty and partiality of information. As a result of synergism of fuzzy logic and neural networks theory, many applications exist in which better performance can be achieved through the use of a neurofuzzy system than through the use of a system which is strictly neuro or strictly fuzzy.

Professor Galushkin's monograph has many unique features that in totality make his work an important contribution to the literature of neural networks theory. He and his publisher deserve profuse thanks and congratulations from all who are seriously interested in the foundations of neural networks theory, its evolution and its current status.

Lotfi A. Zadeh
Professor in the Graduate School
Director, Berkeley Initiative in Soft Computing (BISC)
Berkeley University, California
May 2006

Foreword

The human brain is indeed a triumph of nature. Able to process information rapidly and efficiently via a system of neural networks consisting of vast numbers of neurons, the human brain has evolved to enable a greater awareness of itself and its actions within its environment: the mind.

Natural neural networks are highly complex, nonlinear systems with huge degrees of freedom that employ different principles of information processing from those of computers. Consequently, efforts to develop a viable neural networks theory have been avidly pursued by theoretical scientists for more than half a century. These attempts have had varying degrees of success (and failure); however, as more remarkable advances in brain science are made, brain-style computer technology is becoming increasingly more promising.

In the earliest days, neural network theories developed in America, Europe, Russia and Japan independently. While the scientific traditions for each of these regions are significantly different, Russian science is most unique due in part to its years of isolation from the Western world. Its researchers were able to deeply develop their own theories in mathematics, physics, control optimization theory and other disciplines, and results have been outstanding. Russia's contribution to neural networks theory is yet another example.

Professor Galushkin, a leader in neural networks theory in Russia, uses mathematical methods in combination with complexity theory, nonlinear dynamics and optimization, concepts that are solidly grounded in Russian tradition. His theory is expansive: covering not only the traditional topics such as network architecture; it also addresses neural continua in function spaces. I am pleased to see his theory presented in its entirety here, for the first time for many, so that both the theory he developed and the approach he took to understand such complex phenomena can be fully appreciated.

Shun-ichi Amari
Director of RIKEN Brain Science Institute
June 2004, in Tokyo

Foreword

During the Soviet period, academic achievements in the Soviet Union in many fields of science and mathematics (particularly those not bearing directly on defense) became widely known and respected. Many important Soviet advances in physics, chemistry, and mathematics were widely disseminated and objectively evaluated as being of the highest rank.

Contemporaneously, Soviet achievements in academic traditions bearing on space technology and defense were also obviously successful (as evidenced by world-class achievements in these endeavors). However, the details of the discoveries underlying these achievements were rarely communicated, and some were actively suppressed. From before 1960 through about 1990, meaningful contact between Soviet and Western neural network research was all but eliminated. The 1990s were a decade of 'renormalization of contact' that followed this long break.

Unlike their Western counterparts, Soviet researchers never experienced the full brunt of the 'perceptron ice age,' although their funding levels were somewhat reduced, since Soviet research resources were, to a degree, allocated in mimicry of the U.S. research agenda. Thus, as the volume of Western neural network research activity was reduced to a low level during 1969–1986, Soviet research remained active. This long period of isolated Soviet research was highly productive. Many themes and concepts unknown in the West emerged and were extensively pursued. A few of these were seen in the two learning theory books by Y. Z. Tsypkin (one of Professor Galushkin's mentors) that were translated into English and disseminated worldwide in the 1970s. But there was little Western follow-through on this unusual and somewhat alien material.

This book is the long-awaited panoramic survey of the Soviet and Russian neural network intellectual tradition. It is a gold mine of important ideas and powerful results that are not available anywhere else in English translation. The author, Dr. A. I. Galushkin, is the leading Russian neural network expert and has been a leading Soviet and Russian neural network researcher since the 1970s. Throughout this period, Dr. Galushkin has been afforded access to all significant Western neural network publications. As a result, this book is doubly valuable because it is written not just by an expert, but someone who knows, and refers to in his writing, the intellectual traditions of the West.

Readers will find many unfamiliar and yet well-developed topics in this book, often with powerful mathematical results (e.g., learning laws) ready to be coded up and put to work. The book begins with a survey of the field of neural networks from both the implementation and the architectural points of view. This approach to the subject used to be popular in the West, back when "supercomputers" were less powerful and capa-

cious than today's pocket computer. However, just because this approach has fallen out of favor in the West, there is every reason to believe that it will again become dominant as neural networks take on such important future areas as real-time engine control (e.g., for pulse detonation aviation propulsion engines, where every single impulse must be controlled in detail with many variable parameters uniquely set in a few microseconds), real-time millimeter-scale microwave beam control (at 10-nanosecond time scales at ranges of thousands of kilometers from the antenna array for space launches using ground-based energy), and, of course, for large-scale machine intelligence. Thus, a viewpoint that organizes the subject around implementation and architecture is, in reality, very forward-looking and valuable. The portions covering implementation reliability and failure tolerance are similarly important, notwithstanding the current disinterest in these topics in the West.

An important fact about the organization of this book is that the topics that are covered in detail have essentially all found valuable applications (as this is a book on theory, these are not discussed, which is probably to the liking of the Russian Ministry of Defense). Not only are these useful topics, in most cases they are topics that have been less developed in the West. Take forward control, for example. This is NOT just what is often thought of as 'open-loop control.' It is NOT just adaptive-model-reference control. It encompasses these but allows for anticipatory system response prediction so that feedback corrections can begin to be issued long before large errors emerge in the controlled system. Thus, highly sophisticated models with *preemptive* feedback (anticipating and correcting errors before they can occur) can be built using this theory (much along the lines of the most arcane cerebellum-cortex motor control models, but even more general). This theory supports development of controllers for complex environments in which almost every system state is novel and far from any training example (e.g., for walking robots).

Neural Networks Theory is a major contribution to the neural network literature. It is a treasure trove that should be mined by the thousands of researchers and practitioners worldwide who have not previously had access to the fruits of Soviet and Russian neural network research. Dr. Galushkin is to be congratulated and thanked for his completion of this monumental work: a book that only he could write. It is a major gift to the world.

Robert Hecht-Nielsen
Computational Neurobiology, Institute for Neural Computation, and ECE Department
University of California, San Diego

Contents

Introduction 1
I.1 Neural Computers 1
I.2 Position of Neural Computers in the Set of Large-Powered Computing Facilities ... 5
I.3 The Concept of Computer Universalism 8
I.4 Neural Computer Modularity 9
I.5 The Class of Problems Adequate to Neural Computers 10
I.6 Methods of Coefficient Readjustment 12
I.7 Neural Computer Classification 12
I.8 Some Remarks Concerning the Neural Computer Elemental Base 14
I.9 Neural Mathematics – Methods and Algorithms of Problem Solving Using Neurocomputers 17
I.10 About Neural Networks 21
I.10.1 Neural Network Structures 23
I.10.2 Investigation of Neural Network Input Signal Characteristics 24
I.10.3 About the Selection of Criteria for Primary Neural Network Optimization 24
I.10.4 Analysis of Open-Loop Neural Networks 25
I.10.5 Algorithms for a Multivariable Functional Extremum Search and Design of Adaptation Algorithms in Neural Networks 25
I.10.6 Investigation of Neural Network Adaptation Algorithms 27
I.10.7 Multilayer Neural Networks with Flexible Structure 28
I.10.8 Informative Feature Selection in Multilayer Neural Networks 28
I.10.9 Investigation of Neural Network Reliability 29
I.10.10 Neural Network Diagnostics 29
I.11 Conclusions 30
Literature 31
Appendix 31
A.1 Theory of Multilayer Neural Networks 31
A.2 Neural Computer Implementation 32
A.3 Neural Computer Elemental Base 32

Part I · The Structure of Neural Networks 33

1 Transfer from the Logical Basis of Boolean Elements "And, Or, Not" to the Threshold Logical Basis 35
1.1 Linear Threshold Element (Neuron) 35

1.2 Multi-Threshold Logics ... 37
1.3 Continuous Logic ... 38
1.4 Particular Forms of Activation Function ... 39
Literature ... 40

2 Qualitative Characteristics of Neural Network Architectures ... 43
2.1 Particular Types of Neural Network Architectures ... 43
2.2 Multilayer Neural Networks with Sequential Connections ... 45
2.3 Structural and Symbolic Description of Multilayer Neural Networks ... 47
Literature ... 52

3 Optimization of Cross Connection Multilayer Neural Network Structure ... 53
3.1 About the Problem Complexity Criterion ... 53
3.2 One-Dimensional Variant of the Neural Network with Cross Connections ... 54
3.3 Calculation of Upper and Lower Estimation of the Number of Regions ... 55
3.4 Particular Optimization Problem ... 57
3.5 Structural Optimization by Some Main Topological Characteristics ... 60
3.6 Optimization of a Multilayer Neural Network Structure with K_p Solutions ... 64
Literature ... 66

4 Continual Neural Networks ... 67
4.1 Neurons with Continuum Input Features ... 67
4.2 Continuum of Neurons in the Layer ... 68
4.3 Continuum Neurons in the Layer and Discrete Feature Set ... 68
4.4 Classification of Continuum Neuron Layer Models ... 69
4.4.1 Discrete Set of Neurons ... 69
4.4.2 One-Dimensional and Two-Dimensional m_2 Feature Space ... 69
4.4.3 Continuum of Features – One-Dimensional m_1 for Several Channels ... 71
4.4.4 Feature Continuum – Two-Dimensional m_1 ... 72
4.4.5 Neuron Layer with a Continuum of Output Values ... 72
Literature ... 74

Part II · Optimal Models of Neural Networks ... 75

5 Investigation of Neural Network Input Signal Characteristics ... 77
5.1 Problem Statement ... 77
5.2 Joint Probability Distribution of the Input Signal for Two Pattern Classes ... 79
5.3 Joint Distribution Law for the Input Signal Probabilities in the Case of K Classes of Patterns ... 84
Literature ... 87

6 Design of Neural Network Optimal Models ... 89
6.1 General Structure of the Optimal Model ... 89
6.2 Analytical Representation of Divisional Surfaces in Typical Neural Networks ... 90
6.3 Optimal Neural Network Model for Multidimensional Signals $e(n)$ and $y(n)$ ... 110
6.4 A Priori Information about the Input Signal in the Self-Learning Mode ... 113

6.5 About Neural Network Primary Optimization Criteria in the Self-Learning Mode 114
6.6 Optimal Neural Network Models in the Self-Learning Mode and Arbitrary Teacher Qualification 116
Literature 119

7 Analysis of the Open-Loop Neural Networks 121
7.1 Distribution Laws of Analogous and Discrete Neural Network Errors 121
7.1.1 Neuron with Two Solutions 121
7.1.2 Neuron with a Solution Continuum 124
7.1.3 Analysis of a Neuron with K_p Solutions 126
7.1.4 Analysis of a Pattern Recognition System with a Nonlinear Divisional Surface 128
7.2 Selection of the Secondary Optimization Functional 129
7.3 About Selection of the Secondary Optimization Functional in the "Adalin" System 131
7.4 Development of the Secondary Optimization Functionals Corresponding to the Given Primary Optimization Criterion 132
7.4.1 The Average Risk Function Minimum Criterion 132
7.4.2 Minimum Criterion for R under the Condition $p_1 l_1 = p_2 r_2$ 133
7.4.3 The Minimum Criterion for R under the Condition $p_1 r_1 = a = \text{Const.}$ 134
7.5 Neural Network Continuum Models 135
7.5.1 Neural Network with a Solution Continuum – Two Pattern Classes 135
7.5.2 Neural Network with a Solution Continuum – Continuum of Pattern Classes 137
7.5.3 Neural Network with K_p Solutions – K Pattern Classes 138
7.5.4 Neural Network with N^* Output Channels – K_0 Gradations in Each Class 139
7.5.5 Neural Network with N^* Output Channels – Neural Network Solution Continuum 139
7.6 Neural Network in the Self-Learning Mode and Arbitrary Teacher Qualification 140
Literature 141

8 Development of Multivariable Function Extremum Search Algorithms 143
8.1 Procedure of the Secondary Optimization Functional Extremum Search in Multilayer Neural Networks 143
8.2 Analysis of the Iteration Method for the Multivariable Function Extremum Search 143
8.3 About the Stochastic Approximation Method 146
8.4 Iteration Methods for Multivariable Function Extremum Search in the Case of Equality-Type Constraints upon Variables 146
8.4.1 Search Algorithm 147
8.4.2 Analysis of the Matrix of the Second Derivatives of the Lagrange Function 148

8.4.3 Operation Speed Optimization for the Extremum Search Iteration Procedure in the Case of Equality-Type Constraints 148
8.4.4 Optimal Operation Speed under Constraints (8.6) 149
8.4.5 The Case of Constraints of Equality Type That Can Be Solved 149
8.4.6 Iteration Process Stability under Equality-Type Constraints 150
8.4.7 Convergence of the Iteration Search Method under the Equality-Type Constraints 151
8.5 Iteration Extremum Search Methods for Multivariable Functions under Inequality-Type Constraints 152
8.5.1 Conditions of Optimality 152
8.5.2 Algorithm of Extremum Search in the Case of Inequality-Type Constraints 153
8.6 Algorithm of Random Search of Local and Global Extrema for Multivariable Functions 154
8.7 Development of the Neural Network Adaptation Algorithms with the Use of Estimations of the Second Order Derivatives of the Secondary Optimization Functional 155
8.7.1 Development of Search Algorithms 155
8.7.2 One-Dimensional Case 158
Literature 158

Part III · Adaptive Neural Networks 161

9 Neural Network Adjustment Algorithms 163
9.1 Problem Statement 163
9.2 Neuron with Two-Solution Continuums 164
9.3 Two-Layer Neural Networks 167
9.4 Multilayer Neural Networks with Solution Continuum Neurons 169
9.5 Design of Neural Networks with Closed Cycle Adjustment under Constraints upon Variables 170
9.6 Implementation of Primary Optimization Criteria for Neurons with Two Solutions 173
9.7 Implementation of Minimum Average Risk Function Criterion for Neurons with Continuum Solutions and K_p Solutions 175
9.8 Implementation of the Minimum Average Risk Function Criterion for Neural Networks with N^* Output Channels (Neuron Layer) 177
9.9 Implementation of the Minimum Average Risk Function Criterion for Multilayer Neural Networks 178
9.10 Development of Closed-Loop Neural Networks of Non-Stationary Patterns ... 180
9.11 Development of Closed-Cycle Adjustable Neural Networks with Cross and Backward Connections 182
9.12 Development of Closed-Loop Neural Networks in the Learning Modes with Arbitrary Teacher Qualification 183
9.13 Expressions for the Estimations of the Second Order Derivatives of the Secondary Optimization Functional 185
Literature 187

10 Adjustment of Continuum Neural Networks 189
10.1 Adjustment of a Neuron with a Feature Continuum 190
10.2 Adjustment of the Continuum Neuron Layer 190
10.3 Selection of the Parameter Matrix for the Learning Procedure of the Continuum Neuron Layer on the Basis of the Random Sample Data 190
10.4 Selection of the Parameter Matrix $K^*(i,j)$ for the Learning Procedure of the Neuron with a Feature Continuum on the Basis of the Random Sample Data 193
10.5 Characteristic Properties of the Two-Layer Continuum Neural Network Adjustment Algorithm 195
10.6 Three Variants of Implementation of the Continuum Neuron Layer Weighting Functions and Corresponding Learning Procedures 195
10.7 Learning Algorithm with a_{2g} Secondary Optimization Functional (the Five-Feature Space) for the Two-Layer Continuum Neural Network 198
10.7.1 Learning Algorithm for the Second Layer (Feature Continuum Neuron) 198
10.7.2 Learning Algorithm for the First Layer (Continuum Neuron Layer) 199
10.8 Continuum Neuron Layer with Piecewise Constant Weighting Functions 200
10.8.1 Open-Loop Layer Structure 200
10.8.2 Recurrent Adjustment Procedure for the Piecewise Constant Weighting Functions 201
10.8.3 About Matrix $K^*(i)$ Estimation 202
10.9 Continuum Neuron Layer with Piecewise Linear Weighting Functions 202
10.9.1 Open-Loop Structure of the Neuron Layer 202
10.9.2 Recurrent Adjustment Procedure for the Piecewise Linear Weighting Functions 203
10.10 Continuum Neural Network Layer with Piecewise Constant Weighting Functions (the Case of Fixed "Footsteps") 205
10.10.1 Open-Loop Layer Structure 205
10.10.2 Recurrent Adjustment Procedure for Piecewise Constant Weighting Functions with Variable Interval Lengths t_s 205
Literature 206

11 Selection of Initial Conditions During Neural Network Adjustment – Typical Neural Network Input Signals 207
11.1 About Selection Methods for Initial Conditions 207
11.2 Algorithm of Deterministic Selection of the Initial Conditions in the Adjustment Algorithms for Multilayer Neural Networks 208
11.3 Selection of Initial Conditions in Multilayer Neural Networks 212
11.4 Initial Condition Formation for Neural Network Coefficient Setting in Different Problems of Optimization 216
11.4.1 Linear Equality Systems 217
11.4.2 Linear Inequality Systems 217
11.4.3 Approximation and Extrapolation of Functions 218
11.4.4 Pattern Recognition 218

11.4.5 Clusterization 220
11.4.6 Traveling Salesman Problem 220
11.4.7 Dynamic System Modelling 220
11.4.8 Conclusion 221
11.5 Typical Input Signal of Multilayer Neural Networks 221
Literature 222

12 Analysis of Closed-Loop Multilayer Neural Networks 223
12.1 Problem Statement for the Synthesis of the Multilayer Neural Networks Adjusted in the Closed Cycle 223
12.2 Investigation of the Neuron Under the Multi-Modal Distribution of the Input Signal 224
12.2.1 One-Dimensional Case – Search Adjustment Algorithm 224
12.2.2 Multidimensional Case – Analytical Adjustment Algorithm 226
12.3 Investigation of Dynamics for the Neural Networks of Particular Form for the Non-Stationary Pattern Recognition 231
12.4 Dynamics of the Three-Layer Neural Network in the Learning Mode 235
12.5 Investigation of the Particular Neural Network with Backward Connections 239
12.6 Dynamics of One-Layer Neural Networks in the Learning Mode 242
12.6.1 Neural Network with the Search of the Distribution Mode Centers $f(x)$ 242
12.6.2 Neural Network with N^* Output Channels 245
12.6.3 Neuron with K_p Solutions 248
12.7 Two-Layer Neural Network in the Self-Learning Mode 250
12.8 About Some Engineering Methods for the Selection of Matrix Parameters in the Multilayer Neural Network Closed Cycle Adjustment Algorithms 257
12.9 Design of the Multilayer Neural Network for the Matrix Inversion Problem 258
12.10 Design of the Multilayer Neural Network for the Number Transformation from the Binary System into the Decimal One 261
12.11 Investigation of the Multilayer Neural Network under the Arbitrary Teacher Qualification 262
12.12 Analytical Methods of Investigations of the Neural Network Closed Cycle Adjustment 263
Literature 272

13 Synthesis of Multilayer Neural Networks with Flexible Structure 273
13.1 Sequential Learning Algorithm for the First Neuron Layer of the Multilayer Neural Network 273
13.2 Learning Algorithm for the First Neuron Layer of the Multilayer Neural Network Using the Method of Random Search of Local and Global Function Extrema 277
13.3 Analysis of Algorithm Convergence under the Hyperplane Number Increase 280
13.4 Learning Algorithms for the Second Layer Neurons of the Two-Layer Neural Network 283

13.4.1 Condition of the Logical Function $e(\mathbf{y})$ Realizability Using One Neuron 283
13.4.2 Synthesis of a Neuron by the Functional Minimization Method 285
13.4.3 Neuron Synthesis by the Threshold Function Tables 290
13.5 Learning Algorithm for Neurons of the Second and Third Layers in the Three-Layer Neural Network 290
13.6 General Methods of the Multilayer Neural Network Successive Synthesis 292
13.7 Learning Method for the First-Layer Neurons of a Multilayer Neural Network with a Feature Continuum 292
13.8 Application of the Adjustment Algorithm of the Multilayer Neural Networks with Flexible Structure for the Problem of Initial Condition Selection 293
13.9 About the Self-Learning Algorithm for Multilayer Neural Networks with Flexible Structure 294
Literature 294

14 Informative Feature Selection in Multilayer Neural Networks 295
14.1 Statement of the Informative Feature Selection Problem in the Learning Mode 295
14.2 About Structural Methods for the Informative Feature Selection in the Multilayer Neural Networks with Fixed Structure 297
14.3 Selection of the Initial Space Informative Features Using Multilayer Neural Networks with Sequential Algorithms of the First-Layer Neuron Adjustment .. 299
14.4 Neuron Number Minimization 300
14.5 About the Informative Feature Selection for Multilayer Neural Networks in the Self-Learning Mode 302
Literature 302

Part IV · Neural Network Reliability and Diagnostics 303

15 Neural Network Reliability 305
15.1 Methods for the Neural Network Functional Reliability Investigation 305
15.2 Investigation of Functional Reliability of Restoring Organs Implemented in the Form of Multilayer Neural Networks 306
15.3 Investigation of Multilayer Neural Network's Functional Reliability 308
15.4 Investigation of the Neural Network's Parametrical Reliability 309
15.5 Investigation of the Multilayer Neural Network's Functional Reliability in the Case of Catastrophic Failures 317
Literature 318

16 Neural Network Diagnostics 321
16.1 Neural Network State Graph – The Main Notions and Definitions 322
16.2 Algorithm of Failure Localization in the Neural Networks 323
16.3 Algorithm of the Minimum Test Design for the Failures of the Logical Constant Type at the Neuron Outputs 331
16.4 Method of the Neural Network Adaptive Failure Diagnostics 332
Literature 338

Part V · Summary 339

17 Methods of Problem Solving in the Neural Network Logical Basis 341
17.1 Neuromathematics – A New Perspective Part of Computational Mathematics . 341
17.2 Neural Network Theory – A Logical Basis for the Development of the Neural Network Problem Solution Algorithms 343
17.3 Selection of the Problems Adequate to the Neural Network Logical Basis 344
17.4 The General Structure of the Program Package for Problem Solution in the Neural Network Logical Basis 349
17.5 Multilayer Neural Networks with Flexible Structure 350
17.6 Neural Network with Fixed Structure 352
17.6.1 Generation of the Input Signal of the Neural Network 352
17.6.2 The Multilayer Neural Network Output Signal Generation 355
17.6.3 Formation of the Primary Optimization Criteria 355
17.6.4 Selection of the Open Neural Network Structure 356
17.6.5 Remarks about the Selection of the Open Neural Network Structure that is Adequate to the Class of Solution Tasks 356
17.6.6 Remarks about the Activation Function Selection 358
17.6.7 Selection of the Multilayer Neural Network Structure According to its Hardware Implementation Technology 359
17.6.8 Generation of the Secondary Optimization Functional in the Multilayer Neural Networks 360
17.6.9 Generation of the Algorithm of the Search Procedure for the Secondary Optimization Functional Extremum 360
17.6.10 Formation of the Adaptation Algorithms in the Multilayer Neural Networks 364
17.7 Verification of the Adjusted Multilayer Neural Network 364
17.8 Elaboration of the Plan of Experiments 365
17.9 About the Importance of the Unification of Designations in the Process of Synthesis of the Neural Network Adjustment Algorithms 367
17.10 About Myths in Neural Network Theory 368
17.11 Conclusion 368
References 376

Conclusion 377

Literature 379

Author's Publications on Neural Network Theory 381

Index 391

Introduction

I.1 Neural Computers

Neurocomputers are computers of a new class. Their appearance was determined for two objective reasons: first, the principal stages of the development of modern elemental base technology that mainly determines the development of computer architecture, and second, practical requirements to solve specific problems in a faster and more economical manner.

As far back as the 1950s, the main reason for neural computer development appeared to be a development of the threshold logic that was permanently contradistinguished to the classical development of the elemental base on the basis of AND, OR, NOT, etc. This resulted in the implementation of a series of specific problem-oriented and experimental universal neural computers in the 1960s and 1970s. The terms "neural computer" and "neurocomputer" are not associated with any feature or characteristic of the human or animal nervous system. They are associated only with the conditional name of a threshold element with the adjustable or fixed weights that implement an elementary transfer function of the neural cell. A sharp upswing of LSI technological development at the beginning of the 1970s, as well as the implementation primarily of microprocessor chips with the classic computer architecture was realized on the elemental base on the basis of AND, OR, NOT, etc., resulting in the slowdown rather than complete termination of the development of the computing facilities based on the threshold logic elements. The next upswing at the beginning of the 1980s allowed one to redefine the problem of neural computers due to the fact that VLSI technology, rather than LSI technology, allowed one to implement in one or several chips not only a great number of processing neuron elements, but also a whole set of connections between them. This was not possible before. Such a possibility was provided by both electronic as well as optical implementation methods in the middle of the 1980s.

The main idea of the neural computer construction, either a problem-oriented or universal one, is to develop computers in the analog-digital form. In this regard, the "fast" analog part performs multidimensional operations on the threshold basis. The algorithms of the neural network coefficient adjustment are implemented either in the "fast" manner in the analog form, or in the "low-speed" manner in the form of specialized digital circuits emulating neural algorithms, or in the "low-speed" manner in the digital form, for example, using the universal personal computer.

The development of neural computers requires a design of the principally new algorithms for the multidimensional solution of problems. The time for the solution of the specific problem, on the one hand, only linearly depends upon the problem

dimensionality and, on the other hand, is determined by the convergence time of the iteration process for the solution in the particular neural network.

The main formal basis for the design of the neural algorithms is a neural network theory [I-1 to I-7]. Let us call the main set of operations that are realized in the process of the algorithm refinement as a "logical basis of the problem". For the majority of problems such a basis is the basis $\{\Sigma\overline{ax}\}$. The following problems require this basis: the problems of vector algebra, Fourier transformation and optimization problems.

A class of tasks including solutions of the ordinary differential equations, Poisson, Euler, Navier-Stokes equations, elliptical equations and so on can be reduced to the aforementioned problems.

A logical basis of the computer system is the main group of operations implemented by the elements of the basic operating device. In the case of classical computers, it is a basis of AND, OR, NOT that forms, first, the level of more complex basis (Sheffer stroke, multiple AND, OR, NOT, etc.), and a macro level, i.e., a level of microdevices. The logical basis of the computer system is not determined by the logical basis of the solved problems but requires an additional application of a rather complex program development system.

In the case of neural computers, the logical basis of the computer system in the simplest case is the basis $\{\Sigma\overline{ax}$, sign$\}$. This basis maximally corresponds to the logical basis of the major solved problems.

When solving problems with the neural computers' logical basis, the basis of the problem is in accordance with the basis of the computer system, and there are no artificial shifts in any direction:

- one such shift corresponds to the problems with the threshold basis, whereas the basis of the computer system is AND, OR, NOT;
- another shift corresponds to the problem with the basis that differs from the threshold one, whereas the computer system is neural.

It is supposed that the accordance between the computer system basis and the basis of the problem provides the highest productivity. This statement is trivial in the case of problem-oriented computers that are designed for the solution of a given specific task. However, it is not trivial for the neural computers that pretend to be called universal at the present time.

All the difficulties of multiextremal search by iteration methods using neural computers are evidently preserved. But these difficulties transfer from the software implementation (von-Neumann computers, computers with SIMD and MIMD architectures) to the software/hardware implementation. In short, an algorithmic kernel of the main array of applied problems can be realized in hardware or software/hardware form with maximal operation speed. The neural computer is a maximally parallelized system for a given algorithmic kernel implementation. The number of operation cycles in the problem solving process, i.e., the number of the adjustment cycles for optimization of the secondary functional in the neural computer, is not determined by the subjective intuition of the circuit designer who distributes the processing among the circuit layers consisting of the Boolean elements. This number is also not determined by the subjective views of the programmer who organizes the interaction between layers. On the contrary, it is determined by the physical entity and complexity of the problem.

Homogeneous neural networks possess properties of gradual degradation when some of its elements break down. This phenomenon was noticed by Rosenblatt while designing the three-layer perceptron with the arbitrary connections in the first layer. An excessive number of elements in the first layer were taken. In this case, a function implemented by the neural network is quasi-distributed along the structure. Neural computers are the first example of an analytically calculated computer structure rather than an empirically designed structure based on some subjective views about the problem and the elemental base.

The neural computer implementation methods are mainly divided into three classes:

1. Software emulation of neural algorithms with the help of computers with SISD architecture (for example, classical personal computers), SIMD architecture (for example, Connection Machines) or MIMD architecture (for example, transputer networks);
2. Software/hardware emulation of neural units on the digital elemental base that provides an accelerated performance for the array of operations in the threshold basis and, first of all, such operations as multiplying and addition (Weitek processors, signal processors of the TMS32020 type, etc.);
3. Hardware implementation of the neural unit on the elemental base that is characteristic of the neural algorithms (CMOS-neurochip, optically controlled transmission-type indicators, holographics, etc.);

The efficiency of the neural algorithm implementation for specific problems increases from variant 1 to variant 2 and further to variant 3.

In the case of hardware implementation of the neural unit or software/hardware emulation, the neural computer represents a classical structure of the problem-oriented computer with SIMD architecture. In more complex cases, it represents a mixed MSIMD architecture. Then the computer is a network of asynchronously operating computer units, each one connected with a unit of synchronously operating elements that implement a part of the neural algorithm.

Supercomputers, such as CRAY XMP, and CYBER 205 had enormous computer capacity. However, they were very expensive and their architecture was not in accordance with principles of the neural processing. The high processing capacity was achieved due to the array processors, special pipelined processors, and reduced cycle time. As a rule, such supercomputers were constructed on the basis of modern technology, and they reached a point of their development when further increase of operation speed was restricted by physical limitations for signal propagation in computer circuits. The outcome of this "technological" dead-end was achieved by the use of fine-grained processing that was implemented by computers with SIMD and MIMD architectures of Connection Machine type, Intel Hypercube type, Ncube type, Meiko Computing Surface type and so on. In particular, this outcome was achieved by the use of "neural" processing.

Different approaches for the revelation of parallelism and its implementation on the parallel processors exist:

- Parallelism of sub-problems or events;
- Explicit algorithmic parallelism;
- Geometric parallelism.

The first type of parallelism is efficient for the program that is executed many times with different parameters. It is desirable in this case to represent this program with a set of independent sub-problems (each neuron with its own parameters) and to perform the sub-problems in parallel on different processors.

The array processor for floating-point multiplication is an example of algorithmic parallelism implementation. Organization of parallel processing in this case consists in the appointment of different sub-problems for the pipeline of processors, each of which performs some operation over the data and transfers the result further. In order to provide the efficient performance of the pipeline, one must balance the processors' loading and access and take into account the time required for data transfer from one processing stage to another.

Geometric parallelism consists in data decomposition among processors in such a way that all the required data are located in the random access memory of one processor or also closely located processors. This geometric parallelism is a standard approach that is used for the computers with SIMD architecture. The practical experience shows that a lot of calculations required for the neural network modeling are local. This fact makes real the neural computer implementation related to the geometric parallelism. That is why neural computers with hardware or software/hardware implementation of the neural unit are related to the classes of SIMD or MSIMD architecture, as mentioned above.

A shining example of mathematical operation, in which parallel implementation corresponds to the processes taking place in the neural network, is the operation of multiplication of matrix by vector. In this case, the vector represents an input signal for a one-layer neural network, and the matrix represents coefficients of the layer neurons. The output neural network signals represent a result of multiplication and nonlinear transformation.

Neurocomputers are the object of interdisciplinary research. Consequently, definitions of the neurocomputer can be made only against the background of some more definitions that are adequate to the different branches of science.

Mathematical statistics. Neurocomputers are systems that allow one to form descriptions of stochastic processes and their assemblies that possess complex and often multimodal or a priori unknown distribution functions.

Mathematical logic and automata theory. Neurocomputers are systems in which the algorithm of the solution is represented by the logic network of a particular form, namely by neurons with the complete elimination of Boolean elements of the AND, OR, NOT types. As a consequence, specific connections between elements are introduced.

Threshold logic (1950s and 1960s, computers on the base of threshold logic). Neurocomputers are systems in which the algorithm of the solution is represented in the form of the network with threshold elements with dynamically tunable coefficients and adjustment algorithms that are independent of the input space and network dimensionalities.

Practically all the approaches based on the threshold logic have limitations similar to the Boolean elements. They depend on the network dimensionality and on the input space of elements. It takes place in spite of the fact that these approaches possess external evidence of the neural networks.

Control theory. The complexities of nonlinear dynamic control system synthesis are well known. In the case of neural computers, these complexities are partially overcome when a special case of a control object is taken. This object is well formalized and represents a multilayer neural network. A dynamic process of its adjustment represents a solution. Practically all synthesis methods for adaptive systems of control are transferred to the neural networks in the form of this special case of the control object.

Computational mathematics. Neural computers implement solution algorithms in the form of neural networks. This provides the development of algorithms that are potentially much more parallel than any physical implementation. A set of neural network solution algorithms represents a new perspective field of computational mathematics conditionally called neural mathematics.

Computer engineering. From the viewpoint of computer engineering, the neural computer is a computer system with MSIMD architecture with the following three principal technology solutions:

- The processor element of the uniform structure is simplified up to the level of a neuron;
- Connections between elements are very complex;
- Programming of computational structure is transferred to the adjustment of connection weighting coefficients between processor elements.

A general definition of the neurocomputer. The neurocomputer is a computer system with hardware and software architecture that is adequate to the algorithm execution presented in the neural network logical basis.

I.2 Position of Neural Computers in the Set of Large-Powered Computing Facilities

One can conditionally divide the large-powered computing facilities into two classes: "large-grained" (consisting of a small number of processors) and "fine-grained" (consisting of hundreds and thousands of processors) ones. The neural computers relate to the fine-grained class.

The following class of computers is possible to consider as a new class of computing facilities. These are computers that solve a rather wide range of universal problems and require, as compared with the computers of traditional types, a new hardware implementation, new program development systems and solution algorithms.

According to the given definition of the large-grained processing computer of CRAY type, the computer class "Elbrus" represents a new class of computers as compared with von Neumann computers. And the fine-grained processing computers with SIMD (Single Instruction – Multiple Data) architecture represent a new class of computers as compared with the large-grained processing computers.

Each of the following computer classes represents a new class with respect to the previous one:

- Computers with von Neumann architecture;
- Large-grained processing computers of CRAY type such "Elbrus" computers;
- Fine-grained processing computers with SIMD architecture;
- Fine-grained processing computers with MIMD (Multiple Instruction – Multiple Data) architecture;
- Fine-grained processing computers with the mixed architecture of MSIMD type (Multiple variant of SIMD);
- Neural computers.

Computers with the SIMD architecture were the first fine-grained computers out of four such computers shown in Fig. I.1 in their evolutionary development. They were first designed in the middle of the 1970s. The processor in this case is a single-bit processor with some local memory. This resulted in the necessity to organize a synchronized performance of a set of such units in the process of sufficiently complex solutions.

The development of VLSI technology at the beginning of the 1980s gave rise to the construction of the first production prototypes of the fine-grained computers with MIMD (Multiple Instruction – Multiple Data) architecture. These devices were implemented in the form of asynchronous networks in the 16- and 32-bit microcomputer, including transputer networks.

Several designs of computers with mixed architecture were developed from 1985-1988. They consisted of the kernel in the form of the network of asynchronously functioning processors. Each processor of such a kernel controls a synchronously functioning network of the usual processors with local memory. Such an architecture is sometimes called a multiple variant of SIMD (MSIMD).

The MSIMD architecture is a repetition of the attempt for the algorithm parallelizing on the basis of the network of synchronously functioning processors that is performed on the new qualitative level (MIMD). Such an attempt has been already implemented on the basis of SIMD structures.

In general, neural computers can be regarded as a particular case (or further development) of the computers with MSIMD architecture in which a synchronized unit at each of the asynchronously functioning "fine grain" computers represents not a simple "vulgar" network of single-bit processors with memory, but a meaningful synchronized unit performing a hardware/software (better pure hardware) emulation of the neural algorithm. In the simplest case, such an algorithm represents an operation of multiplying a large dimensionality vector or matrix by a vector. But this occurs only in the simplest case. The neural computers represent a special case of the MSIMD structure in which a synchronously functioning "cluster" of single-bit processors has a special organization that is close to the hardware/software implementation of the main



Fig. I.1. Evolution of fine-grained computer development: SIMD – Single Instruction – Multiple Data; MIMD – Multiple Instruction – Multiple Data; MSIMD – Multiple variant of SIMD

part of the algorithm. In the considered neural case, such a "cluster" represents a hardware/software implementation of the "neural" kernel of a number of algorithms.

The hardware/software implementation of the neural algorithms for the synchronized units of the neural computers most probably also provides the solution of two additional problems:

1. To minimize (or sometimes to eliminate completely) the information interchange between the nodes of the neural computer asynchronous kernel in the process of problem solving. Such a possibility is practically excluded for the majority of problems in the transputer networks or similar systems.
2. To solve so-called weakly formalized problems, such as learning for the optimal pattern recognition, self-learning (clusterization), etc.

It must be mentioned that at present, a supercomputing level is the most important application of neurocomputers out of any other possible applications. This level is characterized by the lack of a computing facility capacity under the existent limitations. In this case, the objective necessity for the neural computer architectural development in the form of the natural development of the MIMD architecture can be also explained by the fact that the natural desire of a design engineer to increase the capacity of the node or to increase the number of nodes in the MIMD architecture leads to the "dead state region" (Fig. I.2). According to the price-capacity ratio criterion, the "dead state region" appears for the following two reasons:

- Objective existence of inter-node exchanges and increase of common losses required for these exchanges when the number of nodes increases;
- Increase of the node cost when its capacity increases.

A dead-state character of the desire to increase the capacity of a computer with the MIMD architecture by means of increasing the capacity of the node or by increasing the number of nodes is determined by the fact that the capacity increases more slowly (or significantly more slowly) than the system cost. The reason for this observation is the higher rate of the node cost increase as compared to its capacity increase as well as the fact that under the increasing number of nodes, the losses required for the internode information exchanges in the process of problem solving also increase.

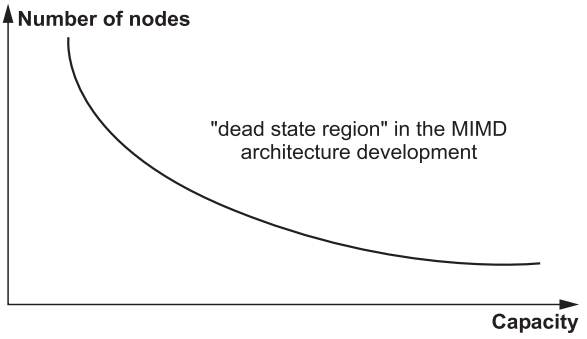


Fig. I.2.
Illustration of limitation of the MIMD architectural development when the node capacity or the number of nodes increases

We consider that namely this is the main reason for the use of some synchronously functioning neural unit in each node of the MIMD architecture. Such a unit implements the neural network in the hardware/software (purely hardware in future) form. In turn, this neural network implements some given function.

Economic factors play and will play a significant role in the process of fine-grained super-computer development, especially in the development of computers with a great number of nodes. The cost increase of the transputer-like custom-designed LSI related to its complication and increase of capacity will determine its use not as a base element of a large array but as a base element of a "middle-size" commutation array. Hence, each node of the latter array is connected with some unit consisting of more uniform, i.e., less expensive, LSIs. These LSIs constitute some co-processor that is problem-oriented either by its structure (a network of single-bit processors with memory) or by its function (for example, a neural calculating machine).

We consider therefore that the role of custom-designed LSI-transputers and transputer-like elements with a structure complication and an enlarged number of transistors will decrease in the domain of computer facility development that corresponds to super-computers with fine-grained structure. The main tendency will consist in the massive usage of the aforementioned elements in the class of personal computers (computer cards, accelerators) and super-personal computers (blocks of several cards for personal computers). They will be also used to a lesser degree in the class of super-mini computers (a column consisting of several dozen cards and several personal computers of Meiko and Megaframe types). The appearance of systems containing several thousands of transputers and transputer-like elements is an objective but temporary phenomenon in the class of super-computers. Moreover, this phenomenon takes place only in the domain of super-computer applications that require unique super-computer samples.

This suggests that in perspective, the development of supercomputers with the fine-grained transputer-type structure containing more than 10 000 nodes is only an uttermost trend. And evolution of this trend can be found in the architecture of a peripheral parts of each node in the commutation array of transputer elements of the future system. Taking into account all the aforementioned remarks, one can conclude that the neural computers represent an effective line for the development of super-computer architectures.

I.3 The Concept of Computer Universalism

Each computer is in some sense problem-oriented to the extent that it solves different problems with different efficiency. However, such specialization becomes less and less expressed in the course of development of each new class of computers, at least due to the enlargement of the application field. One can imagine a qualitative pattern reflecting a degree of universalism of different computer classes at each current time. A possible example is represented in Fig. I.3. At present, neural computers are "more problem-oriented" than transputer-like computers. In turn, transputer-like computers are more problem-oriented than single-processor ones. However, it is a question of time or resources dedicated by perforce to either line of development.

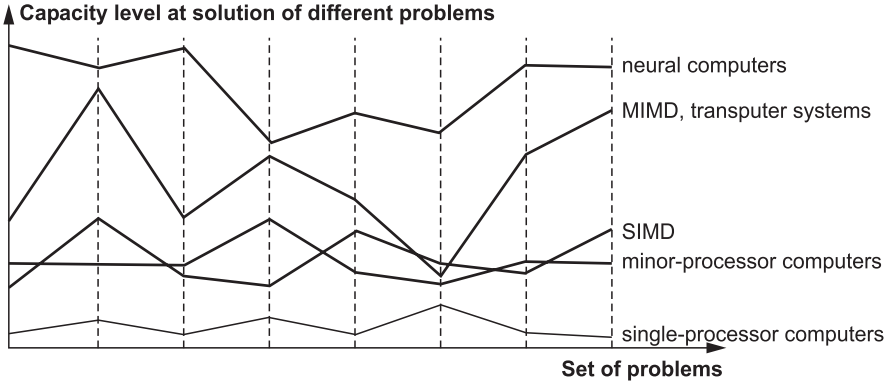


Fig. I.3. Qualitative pattern for the estimation of universalism level for computers of different classes

Fig. I.4.
Illustration of qualitative possibilities of parallelizing presented by computing facilities of different classes

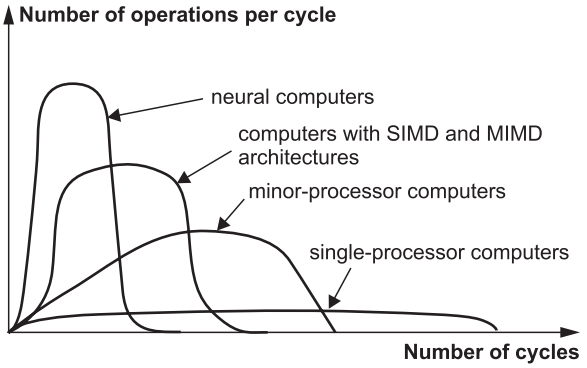


Figure I.4 shows a qualitative illustration of potentialities provided by the parallelizing of algorithms, presented by computing facilities of different classes applied to the solution of some specified problem. The number of operations executed per computer cycle increases due to the increase of potential ability provided by the parallelizing process. As a result, the time required for the solution decreases.

I.4 Neural Computer Modularity

Neural computer modularity (modular extendibility) is determined by the objective requirement for the existence of a transputer or transputer-like kernel in its structure.

Concrete examples of neural computer designs show that the intention to increase their capacity taking into account their real overall-structure characteristics requires the existence of a transputer or transputer-like kernel in their structure. Such a kernel is necessary for the organization of an asynchronous process of information transmission between separate neural units both during preparation for the solution as well as during the solution process. The structure of such a transputer or transputer-like kernel can be different. Sometimes, when the number of asynchronously functioning nodes

is sufficiently large (several hundreds or thousands), it is necessary to transfer from the kernel structure of "lattice" or "torus" type to the structure of "hypercube" type in order to increase the equivalent traffic of information transmission in the kernel.

As distinct from computers with MIMD architecture (Fig. I.2), the capacity of neural computers is uncritical to the capacity of the node computers at the asynchronously functioning kernel. Therefore, there is no need to prove the necessity of capacity increase by means of the obligatory exaggeration of technology-based standards.

It is necessary to notice some properties of the asynchronous kernel of the neural computer. The hardware and software compatibility of neural computers with transputers or with other transputer-like elements is preserved in the kernel construction based on transputers. Neural computers (Fig. I.1) do not represent something exotic, as it was the case in the 1960s. They are a regular result of the evolutionary development of the architecture of fine-grained computers.

Similarly to the transputer systems, neural computers can be used at different levels of implementation:

- Personal computer built-in boards;
- Personal computer units consisting of several boards;
- Columns with control personal computers;
- Assemblage of columns with control personal computers (super-computer level).

Similar to the case of transputer systems, the main goal of neural computer design is the achievement of super-computer or supercomputing level. That is why one must regard the construction of different neural boards for personal computers at present as a pure technological and instrumental stage or a stage of implementation of some insignificant problems. The main strategic problem remains to be the problem of neural super-computer development. As an example, one can consider the development of neural computers at TRW Company (MARK, Input, II, III, IV, V, etc.).

The asynchronous kernel in neural computers will mainly perform two functions:

- Preparation, transmission of initial data and receipt of calculation results from separate neural units at each node of the asynchronous kernel before the beginning of the solution;
- Messaging during the solution process when the problem is insufficiently prepared for the solution by neural computer. It is necessary to minimize the level of such insufficiency and to minimize the messaging traffic in the asynchronous kernel.

I.5 The Class of Problems Adequate to Neural Computers

All the problems solved by computer systems can be conditionally divided into three classes:

1. Formalized;
2. Weakly formalized;
3. Non-formalized.

The class of formalized problems consists of problems with an explicit and transparent solution algorithm that directly indicates the corresponding class of computers and computer architecture (SISD, SIMD, MIMD, etc.) providing the best solution.

The class of weakly formalized problems consists of problems either with a non-unique solution algorithm or with a solution algorithm that does not provide a simple estimation of the solution quality or solution accessibility. Usually the problems with large dimensionality belong to this class. These problems are characterized by so-called "dimensionality damnation" that leads to the necessity to use iteration procedures for their solution with very difficult estimation of iteration process convergence and precision. The iteration procedures adequate to the weakly formalized problems are sometimes used even for the formalized solutions in the case when the formal algorithm is very laborious (for example, for the solution of the linear equation system with sufficiently large dimensionality).

The class of non-formalized problems consists of problems with solution algorithms containing parameters or functions that are implicitly given in the form of description of some input signal class. The examples of such problems are pattern recognition problems, clusterization or self-learning, search for informative attributes, etc.

Notice that in principle, there is a relationship between the aforementioned problem classes and the type of computer architecture. But this relationship cannot be directly expressed. The formalized problems characterized by an essentially consequent algorithm are evidently adequate to the SISD architecture. However, a lot of problems among well-formalized ones can be solved much more efficiently by the use of special parallelization methods. This class of problems represents a sufficiently large domain of the present-day mathematics related to the development of parallelization algorithms. As a rule, the development of a parallelization algorithm for the formalized problem is performed on the basis of the concrete computer architecture (SIMD, MIMD, etc.). The aforementioned weakly formalized problem can evidently be solved using a serial computer, but these problems are more adequate to computers with SIMD and MIMD architectures. Moreover, weakly formalized problems to some degree were the reason for the development of computers with such architectures.

It is possible to consider computers with SIMD and MIMD architectures as devices conditionally adequate to the formalized problems with a sufficiently parallelized solution algorithm and to the weakly formalized problems.

Namely non-formalized problems were the reason for the development of neural computers about thirty years ago, though in principle, the solutions of neural problems were performed on the basis of computers with SISD architecture, and there are attempts to solve such problems on the basis of computers with SIMD and MIMD architectures.

The following main problems must be solved in the process of neural computer development:

- Development of the solution algorithm adequate to the neural computer structure. Selection of a kernel in the solution algorithm structure that is adequate by its structure to the neural network with maximal parallelism (number of neurons in the layers, dimensionality of the attribute space);
- Development of structures and methods for the neural network implementation adequate to the given class of problems;

- Development of neural network adjustment algorithms in the process of solution of given problems and analysis of their convergence;
- Development of the neural network theory sideward universalization of the neural kernel in the enlarged class of algorithms.

The uniform neural network structure for the selection of the solution algorithm in the capacity of the architecture kernel was chosen according to the following reasons:

1. Such a structure provides the possibility of massive parallel synchronous execution of a large number of operations that in turn consist of the simplest operations of addition, multiplication and nonlinear transformations.
2. Such a structure implements sufficiently complex and flexible functional transformation of the input space into the output space.
3. Such a structure enables an analytical description of transformation of the input space of states into the output one.
4. Such a structure enables the organization of the controlled process of the network coefficient adjustment in the adaptive mode.
5. In the future, the use of the linear sequential Gill machines will allow one to come to the solution of the analytical description problem. This in turn will provide the possibility of synthesizing adaptation algorithms in the multilayer neural networks [I-8].

I.6 Methods of Coefficient Readjustment

Methods of readjustment of the neural network weight coefficients in neural computers can be classified in the following way:

- Technological methods (used in the stage of production) similar to those used, for example, in the production of GaAs optical neural chips;
- Schematic design methods (used for the specified user before the exploitation stage);
- System engineering methods (used in the process of functioning) that in turn can be conditionally classified into several subclasses, for example, low-speed methods (as during the solution of linear inequalities) and high-speed methods (as during adaptive processing of the neural network input signal).

I.7 Neural Computer Classification

According to the common opinion of designers, neural networks have a much wider field of implementation than any other implementations of parallelism concepts due to the fact that the property of a large massive parallelism in the case of neural networks is embedded inside of them.

Figure I.5 shows a structure of the main neural computer types. This structure is presented in order to determine the main perspectives of the architectural development for large-powered computers, in particular, for neural computers.

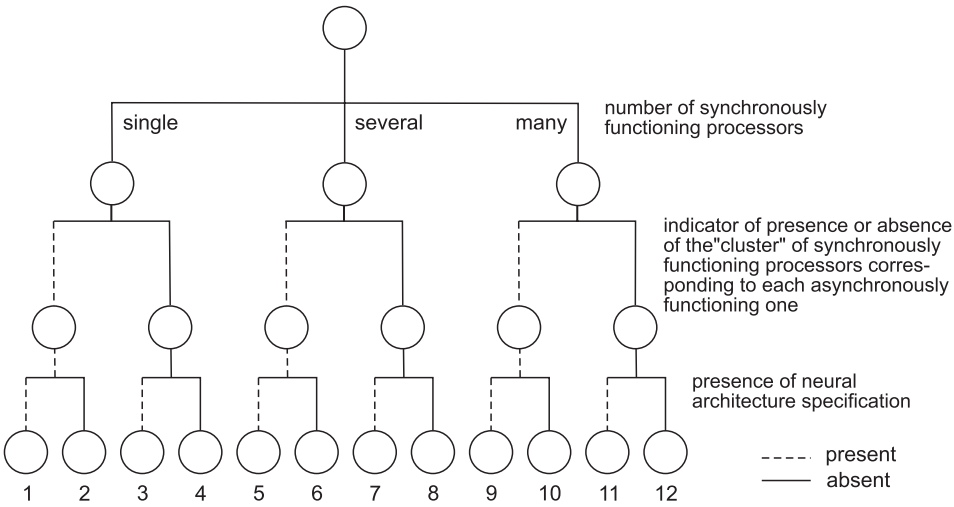

single
several
many
number of synchronously functioning processors
indicator of presence or absence of the"cluster" of synchronously functioning processors corresponding to each asynchronously functioning one
presence of neural architecture specification
present
absent
1 2 3 4 5 6 7 8 9 10 11 12

Fig. I.5. Structure of main types of neural computers

It is assumed that the neural computer capacity increases with the increase of the type number.

Type 1. Computers of this type are represented by the well-known EC computers, CM computers and personal computers.

Type 2. Neural computers of the simplest form in which a neural algorithm is software emulated on the basis of type-1-computers providing an equivalent capacity increase in the process of problem solving.

Type 3. This type of computer includes single-processor computers (large-scale computers, mini computers or personal computers) equipped with array processors. Computers of DAP, IBM with FPS, STARAN, etc. -types can serve as examples.

Type 4. Neural computers with hardware/software emulation of neural algorithms on the basis of type-3-computers.

Type 5. Computers of different classes, beginning with super-computers up to multiple-microprocessor computers, equipped with several numbers of processors (usually not more than 2, 4, 8, or 16). Computers “Elbrus”, WARP, Alliant, etc. are the examples.

Type 6. Neural computers with software emulation of neural algorithms on the basis of type-5-computers.

Type 7. Neural computers with SIMD architecture equipped with several (not more than 2, 4 or 8) command processors.

Type 8. Neural computers of type 7 with the hardware/software implementation neural algorithm of the solution.

Type 9. Computers of transputer type with a large number of processors (dozens, hundreds and thousands).

Type 10. Implementation of neural solution algorithms on the basis of type-9-computers.

Type 11. Computers with MSIMD architecture equipped with a sufficiently powerful transputer or transputer-like kernel. The increased capacity is achieved by the addition of a co-processor to each kernel processor. Thus, each such co-processor represents some unit with SIMD architecture.

Type 12. This computer type is similar to the type-11-computers equipped with synchronously functioning clusters of processor elements implementing neural algorithms.

It is necessary to mention that computers of 4-, 8- and 12-types include a hardware/software implementation of "neural clusters". Such an implementation is significantly oriented namely on the solution based on the neural algorithms with corresponding elemental base, architecture, and program development systems.

From our viewpoint, it is also noteworthy that at present, the computer systems of types 9, 11, and 12 are the most perspective.

In type-11-computers, the disadvantages related to the property 1 are partially eliminated. And in type-12-computers, also the disadvantages related to the property 2 are eliminated.

I.8 Some Remarks Concerning the Neural Computer Elemental Base

The problem of the neural computer elemental base is the most significant one in the determination of the neural computer type that will be designed in the nearest future. Classical technology lines of elemental base elaboration oriented on the neural structures must be developed along with new technology lines appropriate only to the neural computers. Suggestions concerning the development of the perspective neural computer elemental base are formed on the basis of a wide scope of functions. A detailed analysis of the efficiency of problem solving with the use of various technologies will allow one to select in the future the preferences and top-priorities in the neural computer elemental base implementation.

One can consider the following neural computer elemental base as a top-priority one:

- Custom-designed transputer-like 32-bit microprocessor that will allow one to save in the future the already accumulated experience of technological design of transputer and transputer-like systems and to use the already developed reserve of the corresponding software;
- Cascade signal processors of IMS A100- and IMS A110-types;
- VLSIC packages and memory microassemblies with high operation speed and digital capacity of samples;
- Programmable logic IC for the neural processing element implementation.

The following neural computer elemental base can be considered as a second-priority one:

- Custom-designed digital CMOS neural chips;
- Optoelectronic GaAs neural chip;
- Analogous CMOS neural chips.

The following neural computer elemental base can be considered as a third-priority one:

- System of neural units on the slab board;
- Neural blocks based on molecular electronics;
- Quantum neural computers.

A majority of new designs of perspective information processing facilities in Russia and abroad are associated with an obligatory decrease of the requirements of technology standards and an increase of the chip integration level. The concept of neural computers on the one hand provides the increase of the capacity-price ratio during problem solving based on the current technology standards, and on the other hand allows one to use principally new technologies (analogous, optical, charge coupled device, etc.) for the development of high-efficiency systems. We consider that this will allow one to escape difficulties related to the intention to minimize technology standards in the process of digital VLSIC manufacturing.

The importance of microelectronic technologies in computer architectural development is evident. Moreover, we consider that namely technological development gives rise to new types of computer architectures. This was the case in the middle of the 1970s when the appearance of medium-scale IC gave rise to the development of computers with SIMD architecture. Similarly, at the beginning of the 1980s, the development of large-scale IC gave rise to the development of transputers and computers with the MIMD architecture.

Namely the development of microelectronic technologies actualized the active development of neural computers in the second half of the 1980s. A transputer could objectively be designed only after the 32-bit micro-processor, on-chip memory, and channel adapters could be manufactured in a single chip. Similarly, the active development of neural computers started after the hardware implementation of a cascaded segment of the neural network with adjusted or fixed coefficients became possible in a single chip.

It is important to notice that in the case of the neural technology representation of the solution algorithms, one can avoid an abnormal (from our viewpoint) intention towards the "distilment" directed at the submicron technology. Such a trend was characteristic for the computer systems based on the processors of i860, Power PC, Alpha, Mersed, etc. types. The aforementioned intention towards the "distilment" of the submicron technology usually results in the following:

- Short-term, local, and often illusive results in the development of domestic computation facilities based on the use of imported micro-processors;
- Practically zero contribution to the development of domestic microelectronics that constitutes the basis of the future domestic computer science.

Neural computer designs will be efficient by their capacity-price ratio even with the use of available Russian 1.0-micron technology. Such designs will provide a higher-priority development of domestic giant-powered computers.

Design engineers developing a line of highly parallel computers with an increased node capacity must remember that the speed of the information propagation in

the human brain is very low. Therefore, one must think about the development of perspective highly parallel solution algorithms and architectures, including neural ones, rather than about the operation frequency of the node element. It is also necessary to remember that mankind achieved a certain limit that makes it possible to create an engineering system consisting of 3–4 billions of neurons (as in the human brain). However, nobody knows how to organize the system of connections between them.

Neural chip development is one of the main lines in neural computer design. The neural chip structure corresponds to the results of structure and adjustment algorithm development of multilayer neural networks (in the case of commonly used neural computers) and neural network solution algorithms (in the case of problem-oriented and special-purpose neural computers).

However, the development of this technology line will require some period of time during which, for some objects, it will be cost-effective to emulate the neural network solution algorithm on the basis of large-powered computers.

It is necessary to mention a low efficiency of workstation-type single-processor computer applications for the solution of complex problems in the neural network logical basis.

It must be mentioned in addition that in order to emulate neural network algorithms with the use of universal microprocessor facilities, it is more effective to develop architectures oriented on the neural network operation execution than to use standard algorithms oriented on the modification of a single-processor solution.

We consider the following classes of computer facilities:

- Single-processor computers (personal computers, middle-class computers, etc.);
- Small-processor computers;
- Multi-processor computers (computers with massive parallelization, transputer computers, psuedo-transputer computers, computers with transputer kernel and peripheral processors of i860-, Alpha-, Power PC-, etc. types);
- Neural computers.

The priority of Russian computer science belongs to the development of neural computers.

On the modern stage of the development of microelectronic technology and adjoining technologies, neural network technology became adequate not only for the different types of microelectronic and semiconductor technologies but also for optical, optoelectronic, molecular, quantum and other technologies.

It must be mentioned that the appearance of slab board system technology and nanotechnology will result in the development of some new super-parallel architectures. It is clear at present that the neural network architecture technology is adequate to the slab board technology (American and Japanese engineering designs). That is why the attempts to develop functional blocks with old architecture on the level of nano-elements adequate to the single-processor computers can be regarded as dead-end. Starting from nano-neural elements, one can probably achieve principally new architecture elements. It is clear that it will be the elements of super-parallel and large-powered computers.

We consider that the investigation of real structure of biological neural networks aimed at the revelation of their structure peculiarities for its future use in prospective computers is practically useless without software automation technologies for the processing of cytological images. Such technologies must include specially designed software that not only provides fast and high-grade quality input of real images of neural tissue cuts but also allows one to perform the in-line processing of these images.

At present, the neural computer designers are not limited by the properties of real specific neural structures because of their relative simplicity or simplicity of considered problems. The increase in requirements imposed by the used neural structures will cause the development of a neuro-physiology line for their investigation. First of all, this will take place in the domain of different vision and acoustic sensor information processing.

Each new technology gives birth to a new class of architectures of computing facilities. This was the case with SIMD architecture at the end of the 1970s and at the beginning of the 1980s. The same situation was with MIMD architecture at the beginning and at the end of the 1980s. Nowadays it occurs with neurocomputers.

The performance of investigations of real neural network structures aimed at the development of neurocomputers and molecular computers will result in the appearance of original architecture prototypes, because the technological principles of realizing and evolution of biological and molecular neural networks greatly differ from those used at present when realizing VLSI and optical neural networks.

I.9
Neural Mathematics – Methods and Algorithms of Problem Solving Using Neurocomputers

The following question is always actual: what class of problems appears to be most adequate to various computer devices based on some new principles? It was considered for a long time that neurocomputers are efficient at the solution of so-called nonformalized problems or weakly formalized problems. Such problems usually relate to the class of problems that require the inclusion of a learning process based on the real experimental data into the solution algorithm.

At present, this class of problems also includes a second class of problems that does not require the process of learning based on the experimental data. However, this second problem class can be well represented in the neural network logical basis. This problem class is characterized first of all by its pronounced natural parallelization properties in the performance of

- Signal processing;
- Pattern processing, etc.

The use of neural network algorithms will also be efficient in the solution of problems in which the dimensionality of the input information space can be efficiently formed by the use of the Monte Carlo method rather than by the use of standard analytic methods.

We consider that *any problem* can be solved with the help of a neurocomputer much more effectively than with a standard computer due to the fact that any problem

algorithm can be represented in the neural network logical basis with the controlled neural layers number [I-9]. This means that the neural network algorithm for the solution of any problem on the logical level is much more parallel than any of its physical implementation. Notice that in the case of transputer and pseudo-transputer systems, the solution algorithm that is initially less parallel than the physical implementation starts to adapt to a more parallel physical implementation. This property principally differentiates neural computers from such systems as transputer ones or systems with transputer kernel and peripheral processors of i860-type, Alpha-type, Power PC-type, etc. All latter systems are usually characterized by modified algorithms taken initially from single-processor computers in which their designers try to minimize the expenses related to the information exchange between processors in the solution process.

The argument in favor of the viewpoint that neural computers will be more efficient than any other architecture is the great enlargement of the problem class solved at present in the neural network logical basis. In addition to the aforementioned problems, one can also mention

- Solution of linear and nonlinear algebraic equations of high dimensionality;
- Solution of systems of nonlinear differential equations;
- Solution of partial derivative equations;
- Expert systems;
- Solution of optimization problems (linear and nonlinear programming) and other problems.

The transfer to the neural network logical basis in all these problems is usually performed when the dimensionality of the space solution sharply increases or when it is necessary to significantly decrease the solution time.

In general, two sections of neural mathematics are developed: general neural mathematics and applied neural mathematics.

Neural mathematics is one of the fields of computational mathematics in which solutions are performed with the use of algorithms in the neural network logical basis. The main goal of neural mathematic development is to elaborate algorithms with a high degree of parallelism for formalized problems as well as for weakly formalized and non-formalized ones.

A criterion of neural network algorithm efficiency is a decrease in the time required for the solution as compared with traditional methods. The comparison of algorithm efficiency in different neural computer implementations is a separate problem that requires a special investigation.

We shall call a *neural network algorithm* such a computational procedure; the main part of which can be implemented with the help of a neural network. Let us consider P to be a formal problem statement. P includes a set of initial data D and a set of objects R that must be determined. A basis for the neural network algorithm development is a systematic approach in which the problem solving process is represented in the form of some dynamic system functioning in time. Thus, the system input is the data set D, and its output is the set of objects R. The objects of the set R are determined and got their values after the problem solving process.

The development of the neural network dynamic system that solves the posed problem consists of the following stages:

1. Determination of an object that represents an input signal of the neural network. It can be some element of the initial data, some initial value of the determined parameters, etc.;
2. Determination of an object that represents an output signal of the neural network. It can be a solution itself or some of its characteristics;
3. Determination of a desired output neural network signal;
4. Determination of the neural network structure:
   a Number of layers;
   b Connections between layers;
   c Objects representing weighting coefficients.
5. Determination of system error function, i.e., a function that characterizes a deviation of the desired neural network output signal from the real output signal;
6. Determination of a system quality criterion and functional of its optimization, that depends on the error;
7. Determination of weighting coefficient values. This can be done in different ways according to the considered problem:
   a Analytically, directly being based on the problem statement;
   b On the basis of some computational methods;
   c Using a procedure of neural network coefficient adjustment.

A solution with the help of a neural network algorithm consists of the use of a designed computer procedure based on some concrete numerical data values. The solution process includes the following stages:

1. Determination of a specified neural network structure corresponding to the used algorithm;
2. Determination of weighting coefficient values or their direct selection from memory in the case when these coefficients were previously found;
3. Generation of initial parameter approximations, if it is necessary;
4. Transmission of all numerical values to the neural network and activation of this neural network;
5. Neural network functioning according to the following selected modes:
   a A single step mode or a mode with several fixed numbers of steps;
   b A mode with a variable number of steps depending upon the required precision and/or upon the specified numerical values of parameters. In this case, the input signal adjustment process takes place;
6. Obtaining the solution.

In the case of multiple usage of the aforementioned procedures, points (1) and (2) can be performed only once.

We shall call a *neural computer* such a computational system that possesses an architecture providing the aforementioned steps 1–6.

Neural mathematics represents a new field of computational mathematics that is oriented on the design of algorithms for the solution of a wide class of problems with the use of neurocomputers. The suggested approach for the algorithm design includes both well-known computational methods, as well as knowledge already accumulated in the domain of neural network calculations. However, this approach significantly differs from both the first and second methods.

Traditional numerical methods are used in neural mathematics only in the case when these methods can be effectively parallelized and expressed in terms of neural network operations. However, these methods can be sufficiently overworked.

Practically all known approaches for the neural network design relate in general to the selection and analysis of some particular structure forms with known properties (Hopfield, Grossberg or Kohonen networks) or to the analysis of some specific modes of their functioning. The use of neural networks is reduced to the application of these structures for the solution of adequate problems in the case of some modifications of their structural parameters.

The initial point in neural mathematics is the problem statement. This problem statement determines the neural network structure adequate to this problem. If it is necessary to perform some adjustment, then one uses the properties of neural network structure classes that include the obtained structure.

The class of neural network structures is usually sufficiently generalized (multilayer neural networks with sequential cross and backward connections).

As a rule, a neural computer must be oriented towards the fast performance of neural network operations and towards parallelized algorithms of the neural network adjustment.

The development of the neural computer includes the following three parallel lines:

1. Development of the solution algorithms (neural mathematics);
2. Development of the neural network theory, structure classes and methods of their adjustment;
3. Development of the neural computer as an assembly of hardware and software orientated towards the solution of neural mathematic problems.

All these levels of development are connected to each other. On the one hand, the neural network structure for each problem is determined by the problem itself. On the other hand, the development of the neural network theory results in the use of more and more complex neural network structures. On the one hand, the level of the used hardware determines the level of possibilities for neural network and algorithm development. On the other hand, the development of neural mathematics determines the development of neural network theory that in turn develops the hardware implementation.

At present, the line of neural network investigation depends on the line of neural computer development only in the field of software implementation of specific problems and their structures. In the future, both neural computer software and hardware will be determined by the solved problems and by the neural network implementation.

The solution with the help of a neural network computer depends on the adjustment procedure that requires the choice of initial parameter values, the choice of an iteration method step value, etc.

Due to the fact that not all these processes are well formalized and depend on the field of the problem application, they are usually human-aided.

Two of the following types of dialogue are observed in the procedure of the solution:

1. *A dialogue during the process of neural computer preparation for the solution of a specific problem with foregone limitations on the initial data and on the results.* The main part of such a dialogue consists in the adjustment of weight coefficients. After the adjustment procedure, the problem can be solved many times with different initial data and one and the same set of problem parameters and neural network structure. In the case when the weight coefficient adjustment procedure is absent, this type of dialogue is reduced to the selection of the required values out of the neural computer memory.
2. *A dialogue during the process of solution*. It includes the generation of initial value parameters. However, the most labor-consuming part is the dialogue during the process of the input signal tuning. In this stage, a designer can analyze the dynamics of the system quality functional changes and select the step value in the adjustment method.

Such a dialogue is critical for the solution process. That is why in the tasks where the solution time minimization is the most important criterion for the efficiency of the neural network algorithms, this type of dialogue must be minimized by means of its shortening or automation. It can also be completely excluded by means of complete automation of the step value selection or by means of the algorithm complication. In the case when it cannot be done, one can try to design the algorithm without the input signal tuning and with the help of a weighting coefficient adjustment.

I.10 About Neural Networks

A neural network represents a highly parallelized dynamic system with a directed graph topology that can receive the output information by means of a reaction of its state on the input actions. Processor elements and directed channels are called *nodes* of the neural network.

Neural networks at the bottom represent a formal tool for the description of the main part of the solution algorithm based on the neural network. The frame of the present book, as well as the book [I-6], is a system approach to the neural network synthesis, i.e., an approach to the design of the neural networks themselves and adaptation algorithms similar to the classic adaptive control systems.

We consider that four approaches to the neural network investigation are possible:

1. *Psychological approach*, when it is necessary to model some psychological paradigm that requires a development and investigation of the neural network with some definite structure.
2. *Neurophysiological approach*, when the neural network is developed and investigated on the basis of the knowledge about the structure of some brain part. The neural network models functions of this brain part.

3. *Algorithmic approach*, when some mathematical problem is formulated and an adequate neural network with the corresponding algorithm adjusted to this solution is designed on the basis of this formulation.
4. *Systematic approach* that combines all the aforementioned approaches and represents the frame of this book. Figure I.6 shows the general structure of the neural network synthesis basically described in the present study.

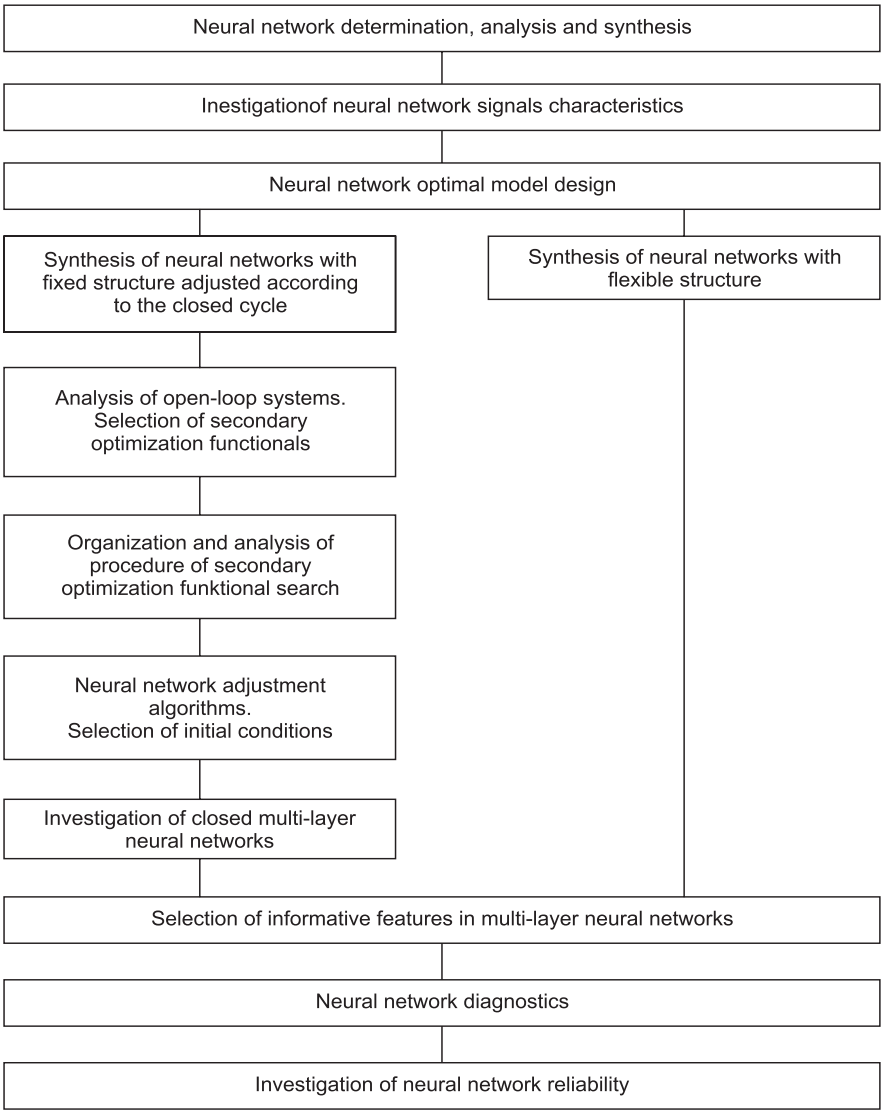

Neural network determination, analysis and synthesis
Inestigationof neural network signals characteristics
Neural network optimal model design
Synthesis of neural networks with fixed structure adjusted according to the closed cycle
Synthesis of neural networks with flexible structure
Analysis of open-loop systems. Selection of secondary optimization functionals
Organization and analysis of procedure of secondary optimization funktional search
Neural network adjustment algorithms. Selection of initial conditions
Investigation of closed multi-layer neural networks
Selection of informative features in multi-layer neural networks
Neural network diagnostics
Investigation of neural network reliability

Fig. I.6. System approach – the multilayer neural network synthesis

I.10.1 Neural Network Structures

The automata theory and the theory of Boolean elements in the 1940s, 1950s and 1960s formed the basis for the development of architecture and separate units of single-processor computers. The same theory of automata based on Boolean elements continues to serve as the logical basis for small-processor, transputer and similar computers, as well as for computers with SIMD architecture in which a network of single-bit processors (STARAN, etc.) represents a peripheral processor.

Similarly, the neural network theory is the logical basis for neural computers. And this has already been the case for several decades (1950s, 1960s and 1970s). At present, this fact has become more evident due to the revolutionary development of the neural computer field.

The neural network represents a network with a finite number of layers consisting of solitary elements that are similar to neurons with different types of connections between layers. The number of neurons in the layers is selected to be sufficient for the provision of the required problem solving quality. The number of layers is desired to be minimal in order to decrease the problem solving time.

This book is dedicated to the description of neural networks with different structures. The objective conditions for the transfer from Boolean to the threshold basis in computer engineering are given. The main types of the threshold elements such as neuron analogues are described. The reasons for investigations of multilayer neural networks are analyzed. The interest in such networks appeared in the 1960s after the publication of the classical work by Rosenblatt [I-1]. Multilayered features are considered specific properties of transformation structure performed by the open-loop system at its topological but not symbolic representation.

Rosenblatt [I-1] investigated multilayer systems with layers consisting of elements with a peer-to-peer topological relationship between the elements of other layers. The layers form sensor elements that represent signal sources for the associative elements of the three-layer perceptron. Associative elements also form a layer of elements whose input consists of the output signals of the sensor elements of the next layer. A multilayer system is a system of elements combined into separate layers with topologically equal properties and different characteristics of connections between the layers of elements. Different types of multilayer neural networks are considered. They include neural networks with sequential connections, cross-connections and backward connection, as well as continual neural networks. Some special types of neural networks are suggested by different authors.

The following main advantages of neural networks as a logical basis for the complex solution algorithms can be mentioned:

- Invariance of neural network synthesis methods upon the dimensionality of the space of features;
- Adequacy of the modern perspective technologies;
- Fault-tolerance in the sense of monotonic but not catastrophic change of the problem solving quality depending on the number of failing elements.

The main goal of this section is to explain why the system aimed at the solution of some specific problem must be designed namely in the form of the neural network and how to choose this neural network topology (the number of layers, the number of layer elements, connections characteristics, topology).

I.10.2 Investigation of Neural Network Input Signal Characteristics

In the first chapter of [I-6], at the investigation of the neural network input signal characteristics in the case of the widely spread task of pattern recognition, a notion of teacher (supervisor) qualification for the input signal distribution functions is introduced. These functions include, in particular, the well-known modes of learning and self-learning. In the general case, the teacher qualification is introduced in different ways for different patterns belonging to objectively different classes. The possibility of introducing more specific input signal characteristics is shown. For example, "teacher's slant about his capabilities" is introduced.

A formal problem statement about neural network learning consists in the approximation of a given sample function of the teacher's instructions by some automatic machine with given properties. A formal problem statement about self-learning is considered a selection in the input signal space of some areas of pattern distribution function modes at the input. A formal problem statement about supervised neural network learning with the teacher of a limited qualifications is a generalization of the first two statements.

The existent investigations in the field of pattern recognition relate mainly to the stationary patterns. In this case, the neural network input signal distribution is time-independent. The present book deals with non-stationary patterns with time-dependent neural network input signal distribution.

I.10.3 About the Selection of Criteria for Primary Neural Network Optimization

A class of statistical theory criteria is usually considered criteria for primary multilayer neural network optimization in the mode of pattern recognition learning. The examples of such criteria are a criterion with a minimum of the average risk function under the condition of equality between the average risk function components for patterns of different classes and a criterion with a minimum of the average risk function under the condition of a given value of the average risk function component for one of the classes.

A precondition for the formation of the criterion and functional for the neural network primary optimization in the learning mode is a representation of the input signal distribution density in the form of a multi-modal function. In this case, some class corresponds to each mode with some probability. Modifications of the average risk function are used at the initial investigation stage as criteria for the primary neural network optimization in the self-learning mode. This criterion requires a natural generalization at the transfer to the continuum of classes and solutions. A separate question considered in the book is the problem of the primary optimization functional formation in the case of arbitrary teacher qualification.

I.10.4
Analysis of Open-Loop Neural Networks

A formal technique used in the analysis of the open-loop systems is based on the precise methods of probability analysis of multidimensional nonlinear systems. The transformation mainly to the analysis of distributions and moments of distribution of errors is related to the fact that the results of such analysis are formally independent of the neural network complexity and type. The exclusion is only characteristic of the feature space and solution space. Further, the latter significant observation is widely used in the stages of selection or the secondary optimization functional formation, as well as during closed-loop neural network development.

The secondary optimization functional is a functional expressed through the parameters of the current signal distributions in the neural networks. It is directly minimized in the multilayer neural networks at the closed-cycle adjustment. At this synthesis stage, mainly two problems are considered.

The first problem concerns the investigation of correspondence between the secondary optimization functionals used in some known works and some criteria of the primary optimization. A matter at issue here is the known adaptive system, such as Adalin, Stainbuh matrix, or Rosenblatt's three-layer perceptron (or rather its adjustable output unit). It is mentioned that the main disadvantage of such approaches is the absence of analysis of correspondence between the used secondary optimization functionals and specific criteria of the primary optimization. This results in the practically complete absence of the operation capability of some systems under the condition of multi-modal distributions of the input signal.

The second problem concerns the formation of the secondary optimization functional corresponding to the given primary optimization criterion. A correspondence in this case is considered in the sense of the coincidence of neural network parameters under the minimum of primary and secondary optimization functionals. The book describes a general method for forming the secondary optimization functional corresponding to the given primary optimization criterion. The results of applying such a method for multilayer neural networks of different structures and for different primary criteria are shown.

I.10.5
Algorithms for a Multivariable Functional Extremum Search and Design of Adaptation Algorithms in Neural Networks

Algorithms for a multivariable functional extremum search and their use for the design of adaptation algorithms in neural networks are described in the sixth and seventh chapters of [I-6].

The problem of a secondary optimization functional extremum search procedure is widely discussed in the literature. We shall mainly consider the aspects of possibility and purposefulness of the use of various gradient procedures (newtonian, relaxation, steepest descent, stochastic approximation, etc.) for the search of local extremum.

The use of iteration methods for the design of multivariable functional extremum search algorithms has some peculiarities at the development of adaptive systems [I-10]. These peculiarities mainly relate to the fact that under the unknown characteristics of the input signal and under the conditions of so-called a priori insufficiency, one cannot say anything about the form of the secondary optimization functional even when the neural network structure is fixed. It is only possible to say about such a property of this functional that it has several local extrema, and all of them, or at least some of them, must be found in the process of closed-cycle adjustment. Namely this fact requires the necessity to introduce some elements of a random nature into the search procedure aimed at the selection of the set of random initial conditions for some gradient procedure. Hence, the main problem is to find the probability of revealing some number of secondary optimization functional local extremums with dependence on the number of emissions of random initial conditions for the gradient procedure of the local extremum search. One of the problems that must be solved in the stage of the closed system design consists in the estimation of the secondary optimization functional gradient vector in the neural networks. This can be done in two ways:

1. Introduction of search oscillations and detection;
2. Representation of gradient vector estimation in the form of expression through the signals in neural networks (output and intermediate signals).

In the first case, one deals with the adaptive search system. In the second case, one deals with the analytical system. It is evident that the design of neural networks in the form of analytical systems adjusted in the closed cycle is preferable because the introduction of search oscillations adds noise to the system. However, the design of closed-loop neural networks using analytical methods is not always possible. The limitations of the analytical approach are shown in the present book while describing the stage of closed-loop neural network design. The main attention in the stage of closed-loop neural network design is paid to the implementation of given primary optimization criteria in the neural networks of different structures.

An important problem is the development of the neural network adjustment algorithms in the self-learning mode and arbitrary teacher qualification mode. The methods of closed-loop system design here are the same as in the learning mode. This is the main idea of the universal approach to the processes of learning and self-learning that constitutes the basis for multilayer neural network synthesis described in [I-6] and in the present study.

The analysis of the known heuristic algorithms for the neural network adjustment based on the idea of a universal approach to the adaptation algorithms synthesis in the neural networks is represented in the work [I-6]. The material concerning adaptation algorithms in continuum neural networks and the selection of adaptation algorithms that are adequate to their physical implementation is also represented in this work.

The peculiarities of the initial condition selection in the neural network adaptation algorithms are also considered.

I.10.6
Investigation of Neural Network Adaptation Algorithms

Investigation of closed-loop neural networks is the final stage of investigation of multilayer neural networks with fixed structure and with closed-cycle adjustment. This synthesis stage deals with some problems related to the open-loop multilayer neural network's performance quality estimation.

The first of these problems is the problem of selection of initial conditions for the adjustment of multilayer neural network coefficients. As it is mentioned above, the secondary optimization functional possesses multiple-extremum properties. This is the reason why two methods of initial condition selection are usually considered: a random method when all local and global extrema must be found and a deterministic method when the multilayer neural network is introduced into the domain of the global secondary optimization functional.

The second of these problems is the problem of selecting a class of multilayer neural network typical input signals that is sufficiently complete to provide the possibility to estimate in the future the neural network's performance quality. In the case of automation control systems, this problem is already solved. In particular, it is solved by the selection of a polynomial input signal class as a typical one. The signal complexity in this case is determined by the polynominal order. In the case of multilayer neural networks, the input signal complexity is determined, in particular, by the modality of conditional distribution of input signals in the task of pattern recognition.

The third of these problems is the problem of selecting optimal parameters for the multilayer neural network tuning circuit. In particular, it is necessary to select a parametric matrix of the system of secondary optimization functional extremum search. Special attention here is paid to the selection of optimal parameters of the neural network tuning circuit based on the estimation of the primary optimization functional value. The results of investigations of a large number of multilayer neural networks obtained by means of computer modeling are presented in the studies [I-1, I-6].

In general, one must take into account the following aspects concerning this multilayer neural network synthesis stage. The consideration of a non-formal problem class of pattern recognition under the unknown and sufficiently complex functions of distribution densities causes difficulties not only for the process of development of such systems, but also for the attempts to estimate theoretically the quality of the solution of these problems. That is why the investigators mainly use methods of statistical modeling. The following results are already published: the results of statistical modeling for separate neurons under the multi-modal distribution of input signals, investigations of adjustment dynamics of three-layer neural networks with sequential connections, and investigations of adjustment dynamics for multilayer neural networks at the non-stationary input signals in self-learning modes and in the supervisor learning mode when the teacher has a finite qualification.

A concrete neural network representing an array of processors with specified connections and implementing adaptation algorithms is a dynamic system which is described by some system of differential or difference equations [I-8, I-11]. A representation of neural networks with closed-loop adaptation circuits in the form of a linear sequential machine makes it possible to describe its functioning in terms of classical

z-transformation. This allows one to use classical methods of statistical dynamics of continuous and discrete automated control systems. This is practically a single efficient way to analyze quantitatively the dynamics of processor array functioning during problem solving. And this way is provided namely by the neural computer's concept. This way makes it possible in the future to analyze and synthesize structures of distributed computers in the form of a processor array from the neural computer class.

I.10.7
Multilayer Neural Networks with Flexible Structure

It has been aforementioned that multilayer neural networks with fixed structure and closed-cycle-adjustment provide the optimization functional optimum under the conditional distribution densities for the probability of the arbitrary input signal that is unknown beforehand. However, the potential quality of such neural networks is limited by the a priori information concerning the structure of the open-loop system. The synthesis methods for neural networks consisting of the open-loop structure part that cannot be fixed a priori and, along with the adjustment coefficients values, represents a result of the adjustment process are considered in the nineth chapter of [I-6] and in the present study. The number of layers and the number of neurons in one layer are determined in the adjustment process. This book also considers some variants of the design of neural networks with flexible structure. The peculiarities of investigation of adjustment procedure dynamics on the level of analysis of optimization functional dependence upon the number of layers and the number of neurons in one layer are described. As a result, the neural network with flexible structure is implemented in the form of the uniform multilayer neural network.

I.10.8
Informative Feature Selection in Multilayer Neural Networks

An attempt to analyze from one viewpoint very different and rather numerous works dedicated to the informative feature selection was done in [I-6]. It was an attempt to develop so-called structural methods related to the methods of multilayer neural network synthesis.

The author considers that the widely spread viewpoint about a possibility of a so-called preliminary feature selection is not valid. The reason is that in the case of any selection features procedure, either directly or indirectly, one must use some concrete neural network. That is why any selection features procedure is subjective. And the subject is the neural network of a specific type.

The second author's proposition in favor of the suggested approach consists in the "absoluteness" of the primary optimization functional in the capacity of the index of features information value. This is the reason that the estimations based on divergence, averaged conditional entropy, etc., are rough and particular.

The aforementioned notions suggest the necessity to analyze the problem of an informative feature selection after the completion of synthesis procedures and after the investigation of the neural network's dynamics. According to the author's opinion, the multilayer neural networks with fixed and flexible structures possess the lowest degree of subjectivity with respect to the input signal (that represents the subject of

investigations with the help of neural networks). The reason is that these neural networks are synthesized under the condition that any information about conditional distribution densities for patterns inside classes is absent. That is why the usual approach is to use the multilayer neural network for the search of maximally informative features of the initial feature space.

The use and investigation of multilayer neural networks allows one to formulate a problem of selection of the most informative features of intermediate spaces, but not of the initial feature space. These intermediate spaces are formed by the output signals of neurons of the first, second and other output layers of the neural network. This problem can be considered the problem of structure minimization (minimization of the neuron number in each layer) for the multilayer neural network after its adjustment coefficient procedure is finished.

I.10.9
Investigation of Neural Network Reliability

The problem of the neural network reliability is at present in the most initial stage of its development. It is evident that its solution will cause a revolutionary influence upon the problem of neural computer implementation on the basis of principally new technologies. In particular, it can be a technology of the slab board system design. The property of the perceptron structure to preserve its functional capability under the breakdown of some number of its elements has been already reported by Rosenblatt in the example of the three-layer perceptron [I-1]. Modern computers do not possess this property because of the absence of explicit reservation. Computers with MIMD (Multiple Instructions – Multiple Data) architecture seem to be the only exclusion. These computers are characterized by the asynchronous principle of the processor's array functioning. This provides so-called gradual system degradation when some of its elements are broken down. The similar property exists in neural computers, too. And this is their great advantage. Methods for neural network functional reliability investigations are represented below. In particular, we describe some experimental methods for functional reliability investigations. In addition, we also describe some methods of parametrical and functional reliability when catastrophic breakdowns occur. Some methods of the development and investigation of restoring organs based on the neural networks are considered separately.

I.10.10
Neural Network Diagnostics

The neural computer's structure is specific to the class of computers designed in the form of a processor array. This asserts some special requirements for the diagnostic procedure in those parts of the neural computer that represent a neural network. The basis for such diagnostic methods was developed in the works [I-12, I-13]. This book describes some methods of forming the notion of neural network failure, some algorithms of neural network failure localizations, methods of minimum test design, and methods of adaptive diagnostics of neural network failures. In principle, the described methods can serve as the basis of development of the neural computer's operation systems for neural network testing.

I.11 Conclusions

As a result of works published in the 1960s, 1970s and 1980s, a line of investigations in the domain of neural network theory that appeared to be prior to the foreign investigations was formed.

The following methods of adaptive neural network adjustment were developed:

- With arbitrary neuron form;
- With arbitrary number of layers;
- With different connections between layers (direct, cross and backward connections);
- With different forms of optimization criteria;
- With different limitations on the neural network weighting coefficients.

The author's basic position in the present study is not the use of the neural network of some given structure that is preferable to some investigator, but the search for the neural network structure and the search for its methods of adjustment adequate to the solved problem.

The content of the book represents the author's current results in the field of neural network investigations. The main line is the development of the neural network theory.

In general, neural computers represent a prospective line of the modern giant-powered computer's development. The neural network theory and neural mathematics represent a foreground line of development of Russian computer science, and they require a support. The bases for the development of these lines are the applied computer systems consisting of neural computers that must be designed in the nearest future.

Development of the following three lines: solutions of neural network algorithms, theory of neural networks and neural computers are tightly interrelated:

- On the one hand, the neural network structure for each task is determined by this task itself, while on the other hand, the neural network theory development provokes the use of more complex structures of neural networks;
- On the one hand, the technology level determines possibilities of neural network and neural network algorithm design, while on the other hand, the development of neural mathematics stimulates the neural network theory; that in turn determines the development of technology;
- At present, the line of neural network investigations is interrelated with the line of the neural computer's development only in the domain of software implementation of the solution of different problems and corresponding structures. In the future, both hardware as well as software components of the neural computer will be mainly determined by solved problems and by neural network structures.

Neural computers represent an efficient symbiosis of computer science, adaptive system automated control theory, and neurodynamics.

The list of additional domestic literature concerning neural computers is given in the appendix to this section.

Literature

[I-1] Rosenblatt F (1962) Principles of neurodynamics. Spartan Books, Washington
[I-2] Minsky M, Papert S (1969) Perceptrons. An introduction to computational geometry. MIT Press
[I-3] Nillson NJ (1965) Learning machines. McGraw-Hill Book Company
[I-4] Grossberg S (1987) The adaptive brain. T.1,2, Advances in psychology
[I-5] Dertouzos M (1965) Threshold logic. A synthesis approach. MIT Press
[I-6] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energiya
[I-7] Ivakhnenko AG (1975) Perceptron – A system of pattern recognition. Kiev, Naukova Dumka
[I-8] Galushkin AI, Fomin Yu I (1991) Neural networks as linear sequential machines. MAI
[I-9] Galushkin AI, Sudarikov VV, Shabanov EV (1991) Neuromathematics: methods of solution using neural computers. Mathematic Modeling, 8
[I-10] Tsypkin YaZ (1968) Adaptation and learning in automatic systems. Moscow, Nauka
[I-11] Gill A (1974) Linear sequential circuits. Analysis, synthesis, and applications. McGraw-Hill Book Company
[I-12] Fomin Yu I, Galushkin AI (1980) Methods of technology diagnostics of threshold element networks. Tekhnika sredstv sviazi, Sistemy sviazy, No. 2
[I-13] Fomin Yu I, Galushkin AI (1981) Methods of failures parallel diagnostics in threshold element networks. Elektronnoye modelirovaniye, 3

Appendix

A.1
Theory of Multilayer Neural Networks

1. Kholmogorov AN (n.y.) Representation of continuous multivariable functions in the form of superposition of several one-variable functions and summation. Doklady AN SSSR 114(5):953–956
2. Glushkov VM (1962) Theory of learning of a special class of discrete perceptrons. Journal of Computing Mathematics and Mathematical Physics 2:317–335
3. Ivakhnenko AG (1963) Can the perceptive self-learning system select transformation moments on the frog retina? Avtomatika 2:31–40
4. Ivakhnenko AG, Kleschev VV, Otkhmezuri GL, Shlezinger MI (1963) Fundamental monograph on the perceptron theory. Avtomatika 3:84–90
5. Glushkov VM (1964) Introduction to cybernetics. Moscow, Nauka
6. Stafford T (1965) Multilayer learning systems. Foreign radioelectronics 8:58–64
7. Gelig AKh (1988) Identifying systems with arbitrary plane retina. Computer Technology and Cybernetics Problems 1988:80–94
8. Galushkin AI (1970) Multilayer systems for pattern recognition. Moscow
9. Galushkin AI, Yumashev SG (1970) Use of piecewise-linear divisional surfaces in the pattern recognition problems. MIEM proceedings 6:238–254
10. Yumashev GG (1970) Use of linear programming methods for construction of piecewise-linear divisional surfaces in the task of pattern recognition. MIEM proceedings 6
11. Ivakhnenko AG (1971) Systems of heuristic self-organization in technological cybernetics. Kiev, Tekhnika
12. Ivakhnenko AG (1971) Polynomial theory of complex systems. IEEE Trans. on System, Man and Cybernetics v. SMC-1, vol. 4.13
13. Galushkin AI (1971) Implementation of primary optimization criteria in pattern recognition systems adjusted by the closed cycle in the learning mode. MIEM proceedings 23
14. Galushkin AI, Vasilkova TA, Slobodenyuk VA, Tyukhov VP (1971) Analysis of dynamics of nonstationary pattern recognition systems. MIEM proceedings 23
15. Agababyan KG (1971) About neural matrices algebra. Dokl. AN SSSR, 199(5):991–993

16. Mkrtchyan SO (1971) Neurons and neural networks. Moscow, Energiya 232
17. Vanyushin VA, Galushkin AI, Tyukhov VP (1972) Design and investigation of multilayer pattern recognition systems. "Some problems of biological cybernetics". In: Berg AI (ed) Leningrad, Nauka
18. Agababyan KG (1972) About neural matrices. Cybernetics 1:211–214
19. Korelov IV (1972) About one class of networks consisting of elements of continuous and discrete performance. Izv. AN SSSR. Ser. "Tekhnitcheskaya kibernetika" 1:109–114
20. Galushkin AI (1973) About adaptation algorithms in multilayer pattern recognition systems. Dokl. AN Ukr.SSR, A, 91(1):15–20
21. Ivakhnenko AG (1975) Perceptrons – systems of pattern recognition. Kiev, Naukova Dumka
22. Galushkin AI, Kudryavtsev AM (1976) Matrix inversion with the help of a multilayer system based on the threshold elements. "Cybernetics and computer technology". Kiev, Naukova Dumka 33
23. Agababyan KG (1976) Perception of angle size by neural structures. Cybernetics 5:173–176
24. Galushkin AI (1977) Continual models of multilayer pattern recognition systems. Automatics and Computer Technology 2:43–48

A.2
Neural Computer Implementation

1. Poupkov KA, Narimanov VK, Galushkin AI (1971) Specialized recognition device. MIEM proceedings, 23:156–165
2. Poupkov KA, Narimanov VK, Galushkin AI (1971) An output cascade of multilayer network of threshold elements. MIEM proceedings, 23:166–178
3. Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energiya
4. Galushkin AI (1977) Main lines of development of specialized multilayer systems for automated medical diagnostics. Medical information systems on a computational basis. Moscow, vol. II, pp 191–198
5. Zaytsev SG (1979) Systems of automated diagnostics based on the use of table data and tendency of their application in medical information systems on a computational basis. Medical information systems on a computational basis. Moscow, vol. II, pp 187–191

A.3
Neural Computer Elemental Base

1. Treyer VV, Elizarov AV (1971) Electrical integrating devices and analogue memory elements. Moscow, Energiya
2. Boyartchenkov MA, et al. (1973) Analogue memory and adaptation elements. Moscow, Energiya

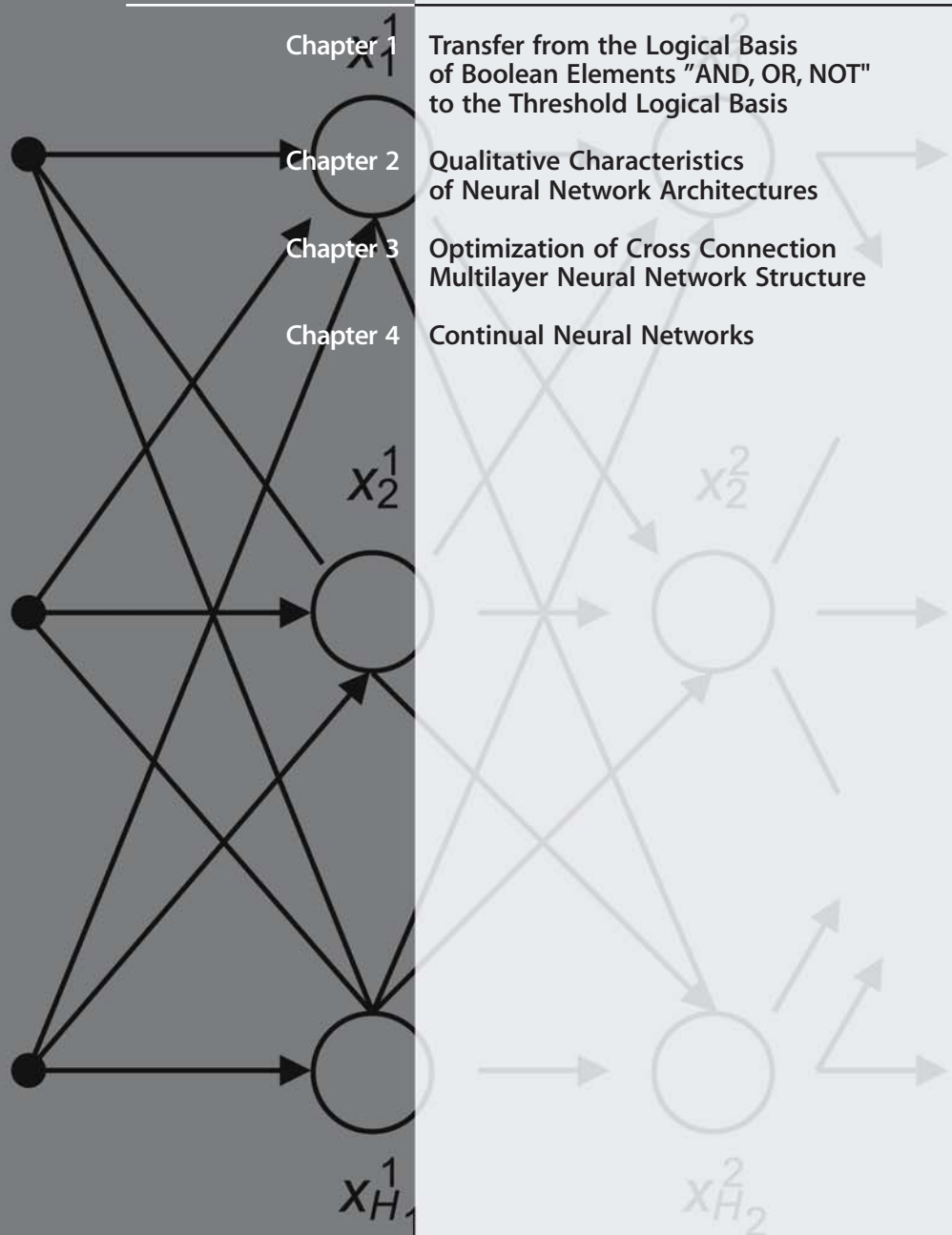
Part I The Structure of Neural Networks

Chapter 1 Transfer from the Logical Basis of Boolean Elements "AND, OR, NOT" to the Threshold Logical Basis

Chapter 2 Qualitative Characteristics of Neural Network Architectures

Chapter 3 Optimization of Cross Connection Multilayer Neural Network Structure

Chapter 4 Continual Neural Networks



The neural network structure is the initial axiomatic base for the neural network theory itself, as well as for the neural network solution algorithms, architecture of neural chips, and neural computer architecture.

We represent below the main reasons determining the transfer from the logical basis of Boolean elements to the threshold logical basis. The main threshold elements are described as well as their relationship with multiple-meaning and continuous logic. The main neural network types proposed by different authors in the 1960s (by Rosenblatt, Widrow, etc.) and at present are described. Materials related to continual neural networks (continuum of features, neurons, etc.) are also presented. We will also separately consider some objective reasons for introducing cross connections in multilayer neural networks, as well as a formal description of methods of such neural networks.

Chapter 1

Transfer from the Logical Basis of Boolean Elements "AND, OR, NOT" to the Threshold Logical Basis

A fundamental point for the neural computer's appearance is the refusal of a Boolean logical basis on the level of computer elements and the transfer to the threshold logical basis. The latter one models some functions of the nerve cell. This transfer changes not only the computer element basis but also all of the computer's architecture.

1.1 Linear Threshold Element (Neuron)

The works concerning threshold logic appeared in the 1960s and 1970s [1-1 to 1-8]. They propose the use of neurons for the design of separate computer units. Here, these neurons perform the following logical transformation of input signals into output ones:

$$y = \operatorname{sign} \sum_{i=0}^{N} a_i x_i \tag{1.1}$$

This is the simplest interpretation of the neuron transfer function. Here, y is a neuron output; a_i are weighting coefficients; a_0 is a threshold; x_i are the neuron input values ($x_i \in \{0,1\}$); and N is a dimensionality of the neuron input signal. The neuron nonlinear transformation in this case is the following:

$$\operatorname{sign}(g) = \begin{cases} 0 \ , & g < 0 \\ 1 \ , & g \geq 0 \end{cases} \tag{1.2}$$

Figure 1.1 shows a functional block diagram of a neuron. In this particular case, when $a_i = 1$ ($i = 1, \ldots, n$), the neuron represents a majority element, and the threshold is equal to $a_0 = N/2$.

A threshold function in the expression for the transfer function (1.1) can take any arbitrary value but is not only determined by expression (1.2). This changes coefficients a_0 and a_i. As a rule, one uses the form (1.2) of the threshold function or the following form:

$$\operatorname{sign}(g) = \begin{cases} -1 \ , & \text{if} \quad g < 0 \\ 1 \ , & \text{if} \quad g \geq 0 \end{cases}$$

The choice depends upon physical implementation of this function either in analogue or digital form.

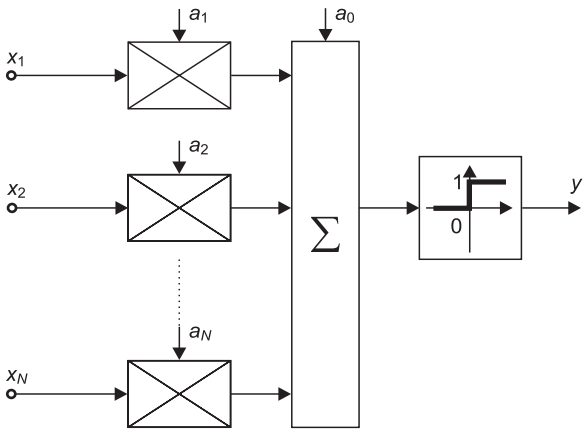


Fig. 1.1.
Functional block diagram of a neuron

The following advantages of a neuron with respect to the Boolean elements AND, OR, NOT, etc., can be mentioned:

1. A neuron performs more complex logical functions. This provides the implementation of a given logical function with the help of a smaller number of elements. As a result, one obtains the possibility to decrease the equipment size during computer unit construction;
2. Neural networks possess a high tolerance against the failure of separate elements. It was mentioned in [I-1] that in the three-layer perceptron, the failure of several numbers of neurons of the first layer with arbitrary connections did not result in the sharp catastrophic decrease of the problem solving quality even in the absence of special diagnostic procedures and structure reconfiguration. Such an ability at present is observed only in multi-microprocessor computers with extended parallelism and SIMD (Single Instruction – Multiple Data) and MIMD (Multiple Instructions – Multiple Data) architectures when diagnostics procedures and reconfiguration are implemented on these structures;

   One of the properties of such structures is the property of so-called permanent degradation. Accordingly, the main quality of system performance (the probability of correct recognition and efficiency) is maximal when all the elements are properly functioning, and this quality decreases when the element failure occurs, but not catastrophically;
3. Neural networks possess an increased tolerance against variation of their circuit parameters. This property can be illustrated by a simple logical function of two variables implemented with the help of a neuron (Fig. 1.2). Rather large variances of weighting coefficients and threshold value do not result in the error of this given logical function implementation.
4. Implementation in the form of VLSI or optical devices with algorithms adequate to neural networks is performed in an analogous form. This form possesses much more operation speed as compared to the digital implementation.

   Maximal information parallelization is achieved when hardware neural network implementation of sufficiently "large" mathematical operations of high dimensionality is realized.

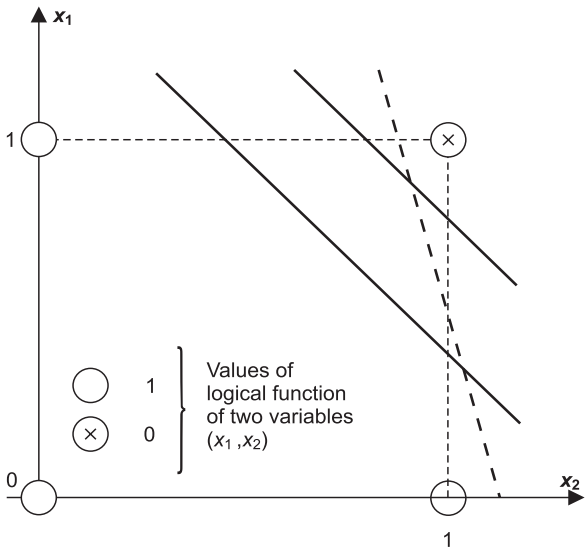


Fig. 1.2.
Neuron tolerance against variation of weighting coefficients and threshold value

5. Neural networks can be formally described as dynamic discrete systems with the use of the linear sequential machine technique. This provides the possibility not only to analyze the behavior of such systems by means of control theory methods, but also to synthesize neural network structures according to the given criteria.
6. It is possible to minimize the VLSI type in the case of uniform neural network design. This allows one to follow up the tendency of simplification of VLSI computer-aided design systems. The relative independency of logical neural VLSI algorithm design upon dimensionality of input and output space forms a basis for standardization of this procedure in a wider class of functional circuits implemented by neural networks.

1.2 Multi-Threshold Logics

Multi-threshold logics can be regarded as the generalization of threshold logics. The logical completeness of a multi-threshold element (MTE) that represents a functional cell of some logical device based on multi-threshold logics is connected with the existence of a group of thresholds implemented by this MTE. We shall consider MTE as an element functioning according to the equation

$$y = \frac{1}{2}\sum_{k=0}^{K}\left\{\operatorname{sign}\left[g(n) - a_k\right] + 1\right\} \tag{1.3}$$

given in [1-6]. The MTE block scheme is shown in Fig. 1.3.

Methods of MTE network synthesis were analyzed during several years [1-7, 1-9, 1-10]. But they did not lead to sufficient results. The most perspective methods for MTE networks development are adaptive methods [I-6, I-10]. These methods are characterized by a weak dependence upon the input space dimensionality and the network complexity.

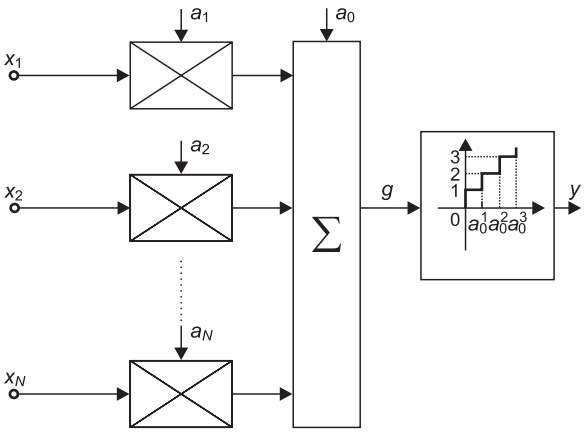


Fig. 1.3.
A block scheme of MTE

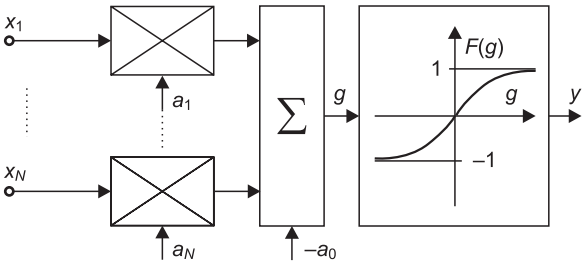


Fig. 1.4.
A block-scheme of a neuron with continuum solutions

1.3 Continuous Logic

This field of science is a natural development of double-digit logic through *K*-digit logic. It has an important significance for the development of neural network computer theory that considers analogous implementation requiring high operation speed.

Continuous neural logic includes circuits that allow one to perform logical operations with continuous variables [1-11 to 1-13]. Figure 1.4 shows a general structure of a neuron with a solution continuum that represents a basis of continuous neural logic circuit design. The behavior of function *f* (that is called an activation function) is considered below according to the neuron model representation or according to the class of solved problems and to the possibility of designing the adaptation algorithm in the most optimal way.

The transformation performed by the circuit shown in Fig. 1.4 has the form

$$y(n) = F[g(n)] = F\left[\sum_{i=0}^{N} a_i x_i(n)\right]$$

The function *F(g)* is a continuous differentiable and steadily increasing function. It forms a continuous output signal of the neural network element. In this case, the notion of a separating surface implemented by the network in the initial feature space degenerates. The parameters of *F(g)* function can be fixed or adjustable.

1.4 Particular Forms of Activation Function

One can consider several forms of activation function in addition to the aforementioned ones.

Activation function *R* is shown in Fig. 1.5. The existence of the linear part allows one to implement continuous activation functions on the basis of such elements. The activation function *R* can be relatively simply implemented.

Activation function *S* (sigmoid function). This function is represented in Fig. 1.6. The expression for this function is $y = (1 + e^{-g})^{-1}$.

As distinct from the activation function *R*, the sigmoid function possesses inversability properties and continuous differentiability properties. But this function is difficult for implementation. Its disadvantage is the existence of only positive values. However, we regard this disadvantage to be insignificant, because it can be eliminated on the level of the network structure by the element threshold change.

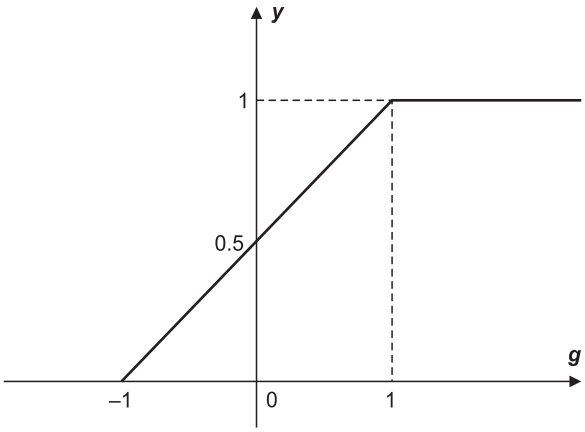


Fig. 1.5.
Activation function *R*

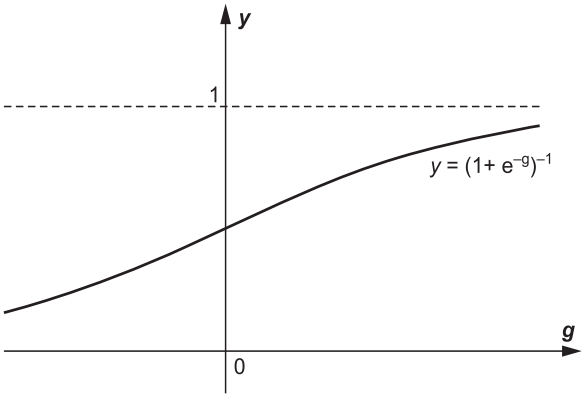


Fig. 1.6.
Activation function *S*

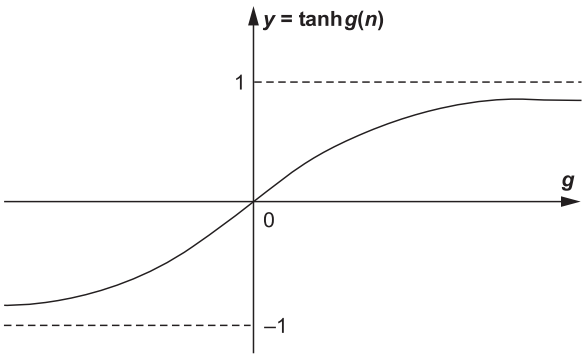


Fig. 1.7.
Activation function tanh (*g*(*n*))

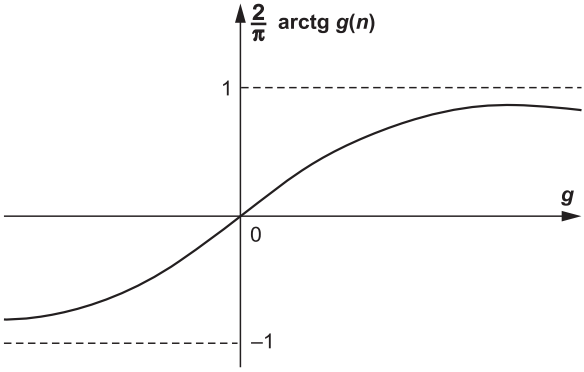


Fig. 1.8.
Activation function
2/π arctg (*g*(*n*))

Activation function tanh(*g*(*n*)) and 2/π arctg(*g*(*n*)) Represented in Fig. 1.7–1.8. These functions are similar by their properties to the sigmoid function. The advantages of the use of any particular activation function are determined by the complexity of their neural network implementation and by their adaptation to the selected class of problems.

Literature

[1-1] Dertouzos M (1965) Threshold logic: A synthesis approach. MIT Press
[1-2] Pirs Y (1968) Design of reliable computers. Moscow, Mir
[1-3] Shigeo Oyagi, Ryoichi Mori, Noriaki Sanechika (1978) Realization of a Boolean function using an extended threshold logic. Bulletin of the Electrotechnical Laboratory 42:9–74
[1-4] Lupanov OB (1972) About threshold element circuits. Dokl. AN SSSR 202(6)
[1-5] Butakov EA (1970) Methods of relay-controlled threshold element devices. Moscow, Sov. Radio
[1-6] Gutchin IB, Kuzitchev AS (1967) Bionics and reliability. Moscow, Nauka
[1-7] Vavilov EN, et al. (1970) Threshold element circuit synthesis. Moscow, Sov. Radio
[1-8] Stepanyan AA, Arkhangelsky SV (1967) Design of threshold element logical circuits. Kuybyshev
[1-9] Minsky M (1967) Finite and infinite machines. Prentice-Hall Inc.
[1-10] Anon. (1988) Specialized issue considering multiple-valued logics. Computer 21(4)
[1-11] Ajzenberg NN, Ivaskiv Yu L (1977) Multiple-value threshold logic. Kiev 148
[1-12] King J (1985) Fuzzy logic provides new way to deal with uncertainty. Electronics 12:40–41
[1-13] Pozin NV (1972) Neural circuits of fuzzy logic and classification of assemblies of analogue signals. Proc. Inst. Stosowane, PAN, Z.II, Warsaw

[1-14] Pozin NV (1970) Neural structure modeling. Moscow, Nauka
[1-15] Winder R (1968) Logical threshold element circuits. Elektronika 11:3–25
[1-16] Butakov VA (1970) Methods for synthesis of threshold element relay devices. Moscow, Sov. Radio
[1-17] Vavilov EN, Egorov BM, Lantsev VS, Totsenko VG (1970) Threshold element circuit synthesis. Moscow, Sov. Radio
[1-18] Varshavski VI (1961) Functional capabilities and synthesis of threshold elements. Dokl. AN SSSR 139(5)
[1-19] Coatec CL, Lewis BM (1964) DONUT – a threshold gate computer. IEEE Trans. on Electronic Computer EC 13(3)
[1-20] Gutchin IB, Kuzichev AS (1967) Bionics and reliability: elements of formal neuron theory. Moscow, Nauka
[1-21] Stepanian AA, Arkhangelski SV (1967) Threshold element logical circuit design. Kuybyshev
[1-22] Lebedev VS (1971) About the possibility of implementation of digital devices using threshold elements. Avtomatika i telemekhanika 3:91–100
[1-23] Mkrtchian SO (1971) Neurons and neural networks. Moscow, Energyia, 231 p
[1-24] Ovchinnikov VV (1971) Method of digital computer arithmetical unit construction using threshold elements. Avtomatika, 6, Kiev, Naukova Dumka, pp 70–77
[1-25] Mkrtchian SO (1972) Complex of threshold elements for the design of digital computers. Voprosy radioelektroniki, Ser. EVT, 2:133–139
[1-26] Pakulov NI, Ukhanov VF, Chernyshov PN (1974) Majority principle of reliable units and device design for digital computers. Moscow, Sov. Radio
[1-27] Dobronravov OE, Ovchinnikov VV (1976) Design of computer circuits and units using threshold elements. Moscow, Energyia
[1-28] Mkrtchian SO (1977) Design of computer logical devices using neural elements. Moscow, Energyia
[1-29] Paljanov IA, Potapov VI (1977) Diagnostics of failures and synthesis of digital structures based on threshold logical units. Novosibirsk
[1-30] Potapov VI (1977) Analysis and synthesis of logical highly reliable digital computational and logical structures based on threshold units. Novosibirsk
[1-31] Galushkin AI, Fomin Yu I (1979) About optimization of restoring organs implementing majority voting. Tekhnika sredstv sviazi, Ser. Vychislitelnaja tekhnika 3:56–61
[1-32] Kirsanov E Yu (1980) About the design of cache memory systems using threshold logic elements. Tekhnika sredstv sviazi, Ser. Systemy sviazi 4:28–37
[1-33] Kirsanov E Yu (1981) About the design of read-only storage using threshold logic elements. Tekhnika sredstv sviazi, Ser. Systemy sviazi 2:44–52
[1-34] Kirsanov E Yu (1981) About a problem of classification and analysis by means of memory design on the basis of threshold logics. Tekhnika sredstv sviazi, Ser. Systemy sviazi 2:36–48
[1-35] Kirsanov E Yu (1981) About the selection of structure of one class of memory units using threshold elements. Elektronnoje modelirovanje, Kiev, 6:88–89

Chapter 2

Qualitative Characteristics of Neural Network Architectures

The main qualitative characteristics of neural network architectures are the following [2-1, 2-5]:

1. Input signal type (dimensionality, discreteness, etc.);
2. Types of operations that are implemented in the open-loop neural network (discrete or continuous);
3. Connection topology (direct, cross, lateral, backward, etc.);
4. Absence or presence of desire to simulate a concrete biological system (visual or acoustic analyzer, cerebellum, thalamus, etc.);
5. A goal to maximally increase the operation speed;
6. Architecture limitations related to the user's convenience or selected technological methods;
7. Method of combining into groups of processor elements;
8. Method of performance in time (discrete or continuous);
9. Method of weighting coefficient modification (random or ordered);
10. Method of connection of independently tuned neural networks.

2.1 Particular Types of Neural Network Architectures

We describe below some particular neural network structures used for the solution of different problems on a neural network basis. Figure 2.1 shows the structure of a so-called neural network with direct connections.

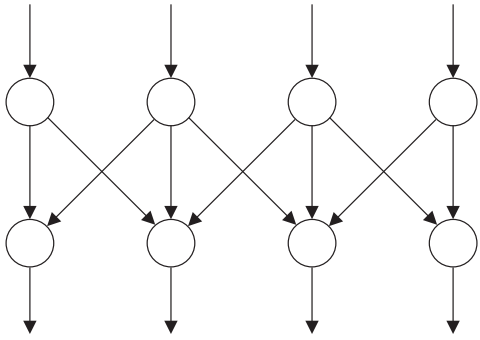


Fig. 2.1. Neural network with direct connections

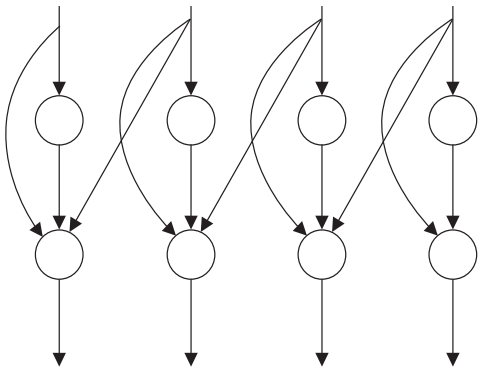


Fig. 2.2.
Neural networks with cross connections

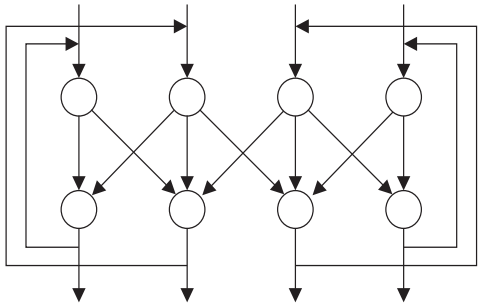


Fig. 2.3.
Neural networks with ordered backward connections

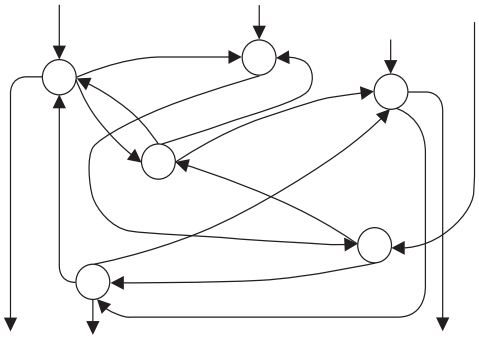


Fig. 2.4.
Neural networks with amorphous backward connections

One characteristic property of such a network is the equality of number of inputs, outputs, and a number of neurons in each of the two network layers. Another characteristic property is the existence of so-called lateral connections between neurons of the first and second layers. Lateral connections in Fig. 2.1 have a limited structure (limited area). Figure 2.2 shows a particular structure of a two-layer neural network with adjustable weighting coefficients of the second layer that are determined by the output signals of the first layer.

Figure 2.3 shows an example of a neural network structure with ordered backward connections, and Fig. 2.4 shows an example of a neural network structure with amorphous backward connections.

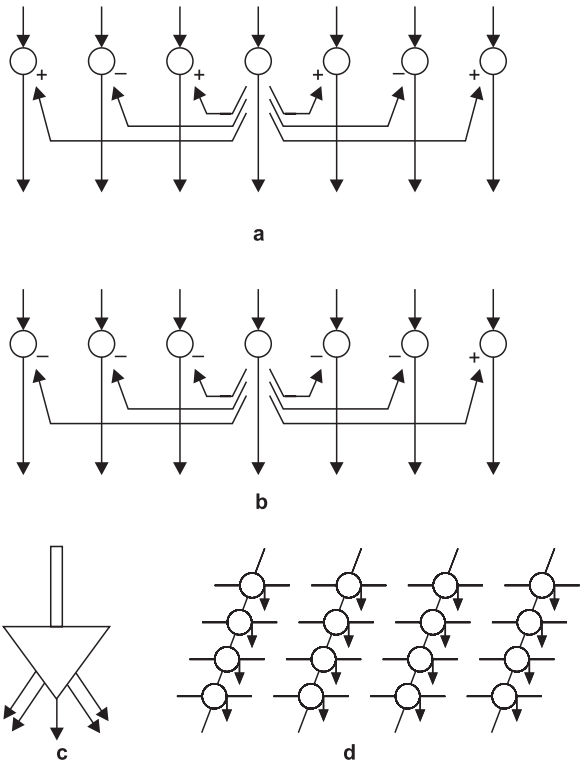


Fig. 2.5.
The main types of neural networks with lateral connections: **a** lateral connections that are repeated on each processor element; **b** lateral connections that are repeated on each processor element; **c** each input has its specific distribution over the field of processor elements; **d** lateral connections can be of any arbitrary type

Figure 2.5 shows particular neural network structures with lateral connections that are most often used in the systems of signals and pattern processing.

2.2 Multilayer Neural Networks with Sequential Connections

Historically, multilayer neural networks appeared in the theory of pattern recognition for the following reasons:

1. The linear divisional surface (linear threshold element) does not provide sufficient probability of correct recognition in the case of distributions that differ from normal distribution with equal covariance matrices.
2. The hyperplane cannot implement any Boolean function of N binary variables in N-dimensional space when $N \geq 2$.
3. In order to increase the probability of correct recognition in the case when two assemblages of vectors of two different pattern classes are distributed according to the law that is more complex than normal distribution with equal covariance matrices, special pattern recognition systems implementing a nonlinear divisional surface are usually constructed. In particular, such a surface can be defined by the expression

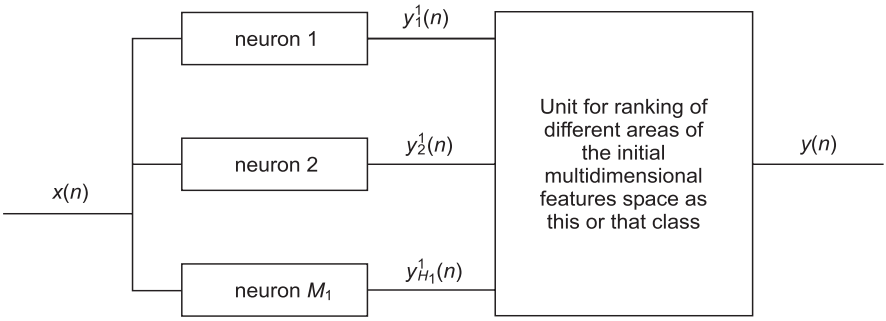


Fig. 2.6. Block scheme of the neural network implementing piecewise linear divisional surface

$$\sum_{l_1=1}^{L}\cdots\sum_{l_r=1}^{L} a_{l_1\ldots l_r} x_{l_1\ldots l_r} + \ldots + \sum_{l_1=1}^{L}\sum_{l_2=1}^{L} a_{l_1 l_2} x_{l_1} x_{l_2} + \sum_{l_l=1}^{L} a_{l_1} x_{l_1} + a_0 = 0$$

Implementation of pattern recognition systems with a nonlinear divisional surface is a complex practical problem. It consists in the adjustment of a large number of coefficients (the approximate order is $(N + \mathrm{r})!/N!\,\mathrm{r}!$ where N is the dimensionality of the feature space, and r is the order of the divisional surface).

For example, when N is about several hundreds and r is about 6–8, then the number of adjustable weighting coefficients amounts to several milliards. The difficulties of the solution to design a nonlinear divisional hypersurface are overcome by the use of its piecewise linear approximation. If a hypersurface of r order is sufficiently well approximated from the viewpoint of the correct recognition probability, then the number of adjustable coefficients is Nr, i.e., in the aforementioned example it amounts to several hundreds. In the task of pattern recognition and in some other tasks with a neural network having neurons in the first layer with a finite number of solutions, one must solve a problem to rank different areas of the initial multidimensional feature space as a particular class (Fig. 2.6).

Such areas appear due to the intercrossing between hypersurfaces implemented by the neurons of the first layer. Each area is determined in the form of set H_1 of binary signals (H_1 is a number of neurons in the first layer) whose values are (0,1) or (+1,–1) at the output of neurons of the first layer and of the corresponding value of the output signal of the whole system. The unit for ranking areas must implement in this specific case some function $y(y_{h1}^1)$ of H_1 binary elements.

This logical function must in turn be realized by the neural network due to the known advantages of the threshold logic and due to the requirement of functional uniformity of the whole system. Figure 2.7 shows a graph of a multilayer neural network with sequential connections.

A detailed classification of multilayer neural networks with sequential connections is given in the work [2-6]. The classification is based on the following features:

- Number of neuron layers with adjustable coefficient;
- Number of neuron layers with fixed coefficients;
- Method of coefficient fixation.

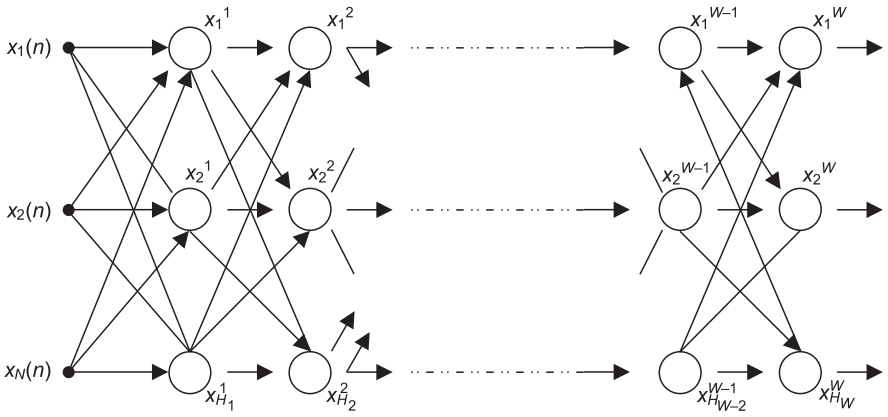


Fig. 2.7. Graph of a multilayer neural network with sequential connections

The following structures are described in particular: a two-layer network with adjustable coefficient of neurons of the first layer and fixed or adjustable coefficients of neurons of the second layer; a three-layer neural network with different variants of layers with adjustable and fixed coefficients.

The two-layer neural network with adjustable coefficients has limited capabilities in the sense of implementation of different configurations of divisional surface because the neuron cannot realize an arbitrary Boolean function of $N \geq 2$ binary variables (in this case, the neuron of the second layer is considered). If limitation on the neuron number of the second is absent, then the implementation of the piecewise linear surface of arbitrary configuration requires not more than three layers in the neural network.

The problem of the three-layer neural network synthesis under the given number of layers in the first and third is reduced to the minimization of the number of neurons of the second layer and adjustment of neural network coefficients. The investigations of multilayer neural networks with sequential connections showed that their performance quality is a monotone increasing function upon the increase of the number of layers and number of neurons in each layer.

2.3 Structural and Symbolic Description of Multilayer Neural Networks

One can mention that recently, the significance of structural methods sharply increased in the investigations of different systems as compared with symbolic methods. The main reasons for this tendency are such properties of these systems as multilayer characteristics, multiple image and high dimensionality. Namely, these properties possess modern neural networks. The present study is aimed at the development of a structural approach. In this approach, the relative role of the learning block development decreases, and correspondingly the role of choice of the open-loop system structure increases. This is the reason that in addition to the open-loop system symbolic descriptions, it is necessary to use the structural representation of transformations. We give below a formal description of the main types of multilayer neural networks.

a Two-layer neural network

$$y(n)=F\left[g(n)\right]=F\left[\sum_{h_1=0}^{H_1}a_{h_1}y_{h_1}(n)\right]=F\left[\sum_{h_1=0}^{H_1}a_{h_1}f\left[g_{h_1}(n)\right]\right]$$
$$=F\left[\sum_{h_1=0}^{H_1}a_{h_1}F\left[\sum_{i=0}^{N}a_i x_i(n)\right]\right] \tag{2.1}$$

b Multilayer neural network of neurons with continuum solution

$$y(n)=F\left[\sum_{h_{W-1}=0}^{H_{W-1}}a_{h_W,h_{W-1}}F\left[\sum_{h_{W-2}=0}^{H_{W-2}}a_{h_{W-1},h_{W-2}}F\left[\sum_{h_{W-3}=0}^{H_{W-3}}a_{h_{W-2},h_{W-3}}x\right.\right.\right.$$
$$x_{h_W}^{W}(n)\quad x_{h_{W-1}}^{W-1}(n)\quad x_{h_{W-2}}^{W-2}(n)$$
$$g_{h_W}^{W}(n)\quad g_{h_{W-1}}^{W-1}(n)\quad g_{h_{W-2}}^{W-2}(n)$$
$$\ldots a_{h_{W-j+2},h_{W-j+1}}F\left[\sum_{h_{W-j}=1}^{H_{W-j}}a_{h_{W-j+1},h_{W-j}}\cdots\right.$$
$$x_{h_{W-j+1}}^{W-j+1}(n) \tag{2.2}$$
$$g_{h_{W-j+1}}^{W-j+1}(n)$$
$$\overset{W-j}{x}_{h_{W-j}}(n)$$
$$\left.\left.\left.\ldots a_{h_2,h_1}F\left[\sum_{h_0=1}^{H_0}a_{h_1,h_0}x_{h_0}^{0}(n)\right]\ldots\right]\ldots\right]\ \right]\ \right]$$
$$x_{h_1}^{1}(n)\quad x_{h_0}^{0}(n)$$
$$g_{h_1}^{1}(n)$$

Here $N=H_0$ is the dimensionality of the initial feature space.

The arrow and symbol indicate the signal designation that is described in the equation by the expression to the right of the arrow, and

$$x_{h_{W-j+1}}^{W-j+1}(n)\quad\text{and}\quad g_{h_{W-j+1}}^{W-j+1}(n)$$

are respectively the analogue output signal and digital output signal of h_{W-j+1}-th neuron of $(W-j+1)$-th layer for the considered multilayer neural network.

A multilayer neural network with K solutions is obtained by means of the exchange of nonlinear transformation f in (2.2) by the expression determined by Eq. (1.3).

c Multilayer neural networks with H_w output channels.

A symbolic description of such a system is relatively simple to obtain from Eq. (2.2) and from the graph of the neural network shown in Fig. 2.7. In particular, one can consider the case of signals $\varepsilon(n)$ and $y(n)$ with equal dimensionalities.

d Multilayer neural networks with cross connections.

In multilayer neural networks with full cross connections [I-6], the set of features of j-th layer ($j = 1,\ldots,W$) consists of features of the initial space and output signals of all layers with numbers from 1 to ($j - 1$).

Analysis of particular structures with full cross connections shows that they are significantly simpler in terms of the number of neurons than those of structures with full sequential connection under the condition when both structures implement the same configuration of divisional surfaces in the feature space. In particular, for the two-layer neural network with cross connections,

$$y(n) = F\left[\sum_{h_1=0}^{H_1} a_{h_1} F\left(\sum_{i=0}^{N} a_{ih_1} x_i(n)\right) + \sum_{j=0}^{N} a_j x_j(n)\right]$$

A graph-scheme of such a neural network is shown in Fig. 2.8.

It is possible in principle to consider a multilayer cross connection neural network of an arbitrary structure.

e Multilayer neural networks with backward connections.

For the neuron with backward connection (Fig. 2.9):

$$y(n) = F\left[\sum_{i=0}^{N} a_i x_i(n) + a' y(n-1)\right]$$

For the multilayer neural network with backward connections (Fig. 2.10):

$$y(n) = F[g(n)]; \quad g(n) = \sum_{h_1=0}^{H_1} a_{h_1} y_{h_1}(n) + a_k y(n-1)$$

$$y_{h_1}(n) = F[g_{h_1}(n)]; \quad g_{h_1}(n) = \sum_{i=0}^{N} a_{ih_1} x_i(n) + a'' y(n) + a'_{h_1} y_{h_1}(n-1)$$

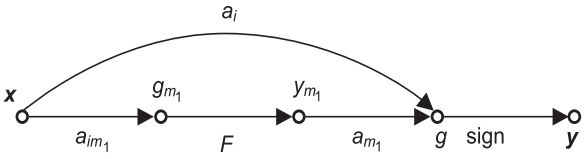


Fig. 2.8.
Graph-scheme of the neural network with cross connections

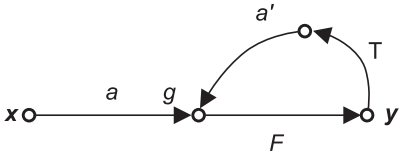


Fig. 2.9.
Graph-scheme of a neuron with backward connection

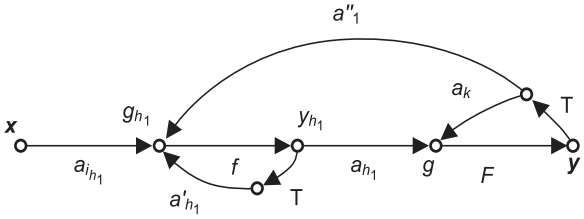


Fig. 2.10.
Graph-scheme of a two-layer neural network with backward connections

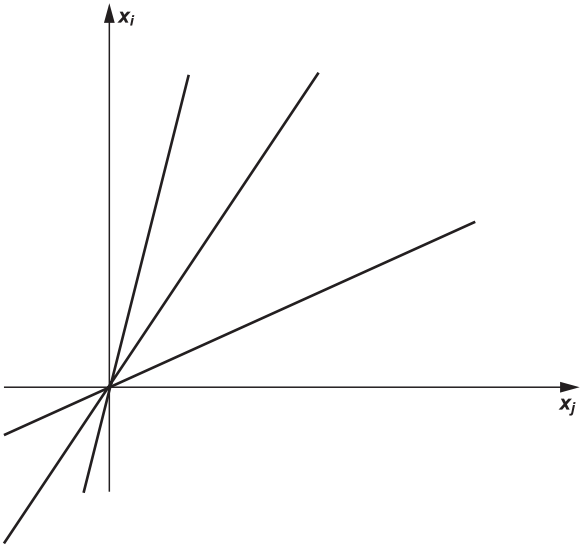


Fig. 2.11.
Hyperplanes realized in the input feature space by the neuron with *k*-valued weighting coefficients

It is possible in principle to consider multilayer neural networks of arbitrary given structure with backward and cross connection. The objective reason for cross connections being introduced in a multilayer neural network is proved below in Chap. 3. As far as backward connections are concerned, they are considered in the present book in the investigation of closed-loop neural networks for nonstationary pattern recognition.

f Neural networks with *k*-valued and binary coefficients.

The difficulties of physical implementation of adjustable variable weighting coefficients of multilayer neural networks are well known. They emerged, in particular, at the development of memistor systems in the 1960s. Their authors tried to implement an open-loop system and block neural network adjustment in the analogue form [2-12].

These difficulties remained the same on the modern stage of VLSI technology. However, the sharp increase of the integration level provides implementation of neural networks with neurons having *k*-valued weighting coefficients realized, for example, on the resistor networks. In the simplest case, the binary values (0,1) of weighting coefficients realized on the monitored switches can be used. This provides sharp simplification of physical implementation of the multilayer neural net-

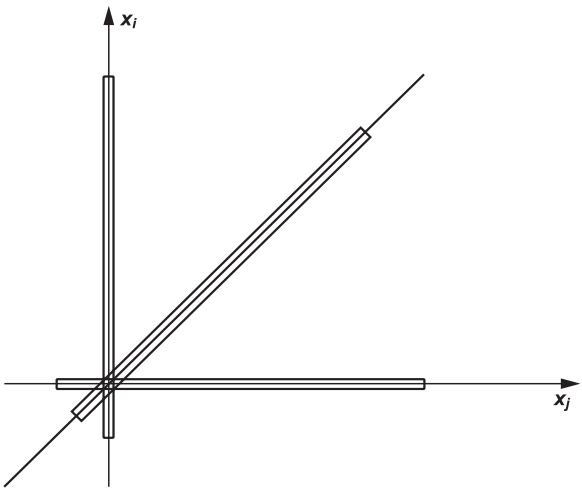


Fig. 2.12.
Hyperplanes realized in the input feature space by the neuron with binary values of weighting coefficients (0,1)

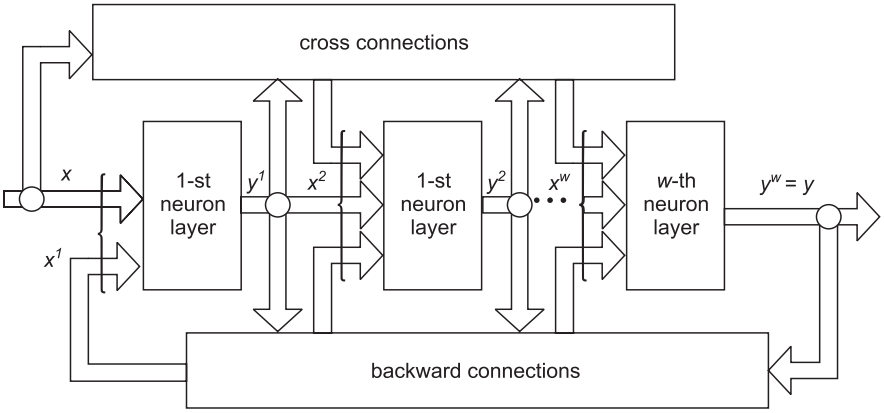


Fig. 2.13. Multilayer neural network

work adjustment procedure consisting of such adjustable coefficients. When considering each neuron with *k*-valued or binary coefficients, a logical function in the input variable space is realized. This is done by means of the divisional hyperplane slope change by some fixed level (Fig 2.11) or by the use of hyperplane "parts" of three types on the total hypersurface (Fig 2.12).

Evidently, the lower the number of gradations the weighting coefficients of neurons in the neural networks have, the higher the number of neurons in the neural network is necessary for the solution of some problem.

The modern level of technological development is quite ready to accept a general neural network structure represented in Fig. 2.13. Methods of adjustment algorithm synthesis for such neural networks are the main subject considered in the present book.

Literature

[2-1] Stafford N (1965) Multilayer learning circuits. Foreign radioelectronics 8:58–64
[2-2] Gavronski R (1968) Comparison of some properties of symmetrical layer networks with forward and backward local connections. Biological cybernetics, pp 209–223
[2-3] Nilsen N (1967) Learning machines. Moscow, Mir
[2-4] Gelig A Kh (1968) Recognition systems with unlimited plane retina. Computer technology and problems of cybernetics, pp 80–94
[2-5] Ajzerman MA, Braverman EM, Roznoer LI (1970) Method of potential functions in the theory of machine learning, Moscow, Nauka
[2-6] Galushkin AI (1970) Multilayer pattern recognition systems. Moscow, MIEM
[2-7] Galushkin AI, Yumashev SG (1970) About the use of piecewise linear divisional surfaces in the pattern recognition problem. MIEM proc. 6:238–254
[2-8] Minski M (1969) Computation: Machines, Prentice-Hall, Finite and Infinite
[2-9] Minski M, Pejpert S (1971) Perceptrons, Moscow, Mir
[2-10] Ivakhnenko AG (1971) Systems of heuristic self-organization in technology cybernetics. Kiev, Tekhnika
[2-11] Ivakhnenko AG (1971) Polynomial theory of complex systems. IEEE Trans. on System. Man and Cybernetics, vSMC-1, vol. 4
[2-12] Trejer VV, Elizarov AB (1971) Electrical integration elements and analogue storage elements. Moscow, Energiya
[2-13] Ivakhnenko AG (1975) Perceptrons – Pattern recognition systems. Kiev, Naukova Dumka
[2-14] Yumashev SG (1970) Use of linear programming methods for the design of piecewise linear divisional surfaces in the pattern recognition problem. MIEM proc. 6:255–260
[2-15] Kohonen T (1980) Associative memory. Moscow, Mir
[2-16] Amari S (1980) Topographic organization of nerve fields. Bull. Of Math Biology 42(3):339–364
[2-17] Fukushima K (1980) Neocognitron: a self-organizing neural network model for a mechanism pattern recognition unaffected by shift in position. Biol Cybern36:193–202

Chapter 3

Optimization of Cross Connection Multilayer Neural Network Structure

The problem of selection of an open-loop multilayer neural network structure is very complex. This structure can be taken a priori, or according to the aforementioned reasons while considering a two-layer and three-layer neural network, or according to the limited technology facilities. We consider below the possibility of the structure selection (number of layers and number of neurons in the layer) for a multilayer neural network with cross connections consisting of neurons with two solutions.

3.1 About the Problem Complexity Criterion

It is necessary to discuss the problem of a complexity criterion for a pattern recognition task solved by the multilayer neural network. The number of reference patterns included in the closed areas by hyperplanes realized by a neuron of the first layer in the initial feature space can serve as such a criterion when a deterministic neural network model is used. In the case of a probabilistic neural network model, each reference pattern corresponds to the mode of distribution probability function for pattern assemblage at the neural network input. In each area of the initial feature space, the multilayer neural network selects some compact pattern set but not the reference pattern. When the assemblage of patterns possesses multimodal distribution at the neural network input, these compact sets can be characterized by some areas in the multidimensional feature space formed by lines of equal distribution probability function values (on the certain level). The number and complexity of such areas characterize together the complexity of the solved problem. The deterministic neural network model can be considered as a particular case of the probabilistic one, and it realizes at the bottom the system of memory for a finite number of multidimensional vectors. The number of areas realized by the multilayer neural network in the initial feature space is considered in this paragraph as a quality criterion of this neural network. The quality of the aforementioned multilayer neural network with sequential connections increases with the increase of the number of layers and the number of neurons in each layer.

Therefore, the problem of neural network optimization (minimization of numbers of layers and neurons) is formulated either to eliminate neuron number redundancy, or under limitation on the number of neurons.

The main attention below is paid to the multilayer neural networks with full cross connections. In this case, a feature set of each j-th layer consists of features of initial space and input signals of the first, second and (j–1)-th layers. The problem of structure optimization for such a neural network is relevant.

3.2 One-Dimensional Variant of the Neural Network with Cross Connections

Let us consider the principle of cross connection operation on the simplest one-dimensional example with $N = 1$ (one feature x). Figure 3.1 shows a block scheme of such a neural network. The divisional surface realized by the network without a cross connection is shown in Fig. 3.2.

In regions I, II and III, analogous output signal of the neural network g under the activated cross connection is represented in the form

$$g_{\mathrm{I}} = a_0 + a_n x - a_1 - a_2$$
$$g_{\mathrm{II}} = a_0 + a_n x + a_1 - a_2$$
$$g_{\mathrm{III}} = a_0 + a_n x + a_1 + a_2$$

The neural network divides each of the regions I, II and III into two subregions, where $g \geq 0$ and $g < 0$. From the condition of zero values of g_{I}, g_{II}, g_{III}, one obtains the expressions for additional thresholds under the activation of a backward connection in the space $\boldsymbol{X}$:

$$x_1 = \frac{a_1 + a_2 - a_0}{a_n} \quad ; \quad x_2 = \frac{a_2 - a_1 - a_0}{a_n} \quad ; \quad x_3 = \frac{-a_1 - a_2 - a_0}{a_n}$$

Thus, the considered neural network (Fig. 3.1) realizes not more than five thresholds that divide the x axis into six regions. The neural network in this case is equivalent to the multilayer neural network with sequential connections consisting of five neurons in the first layer. This means that the neural network with cross connections is realized much more simply than the neural network with sequential connections.

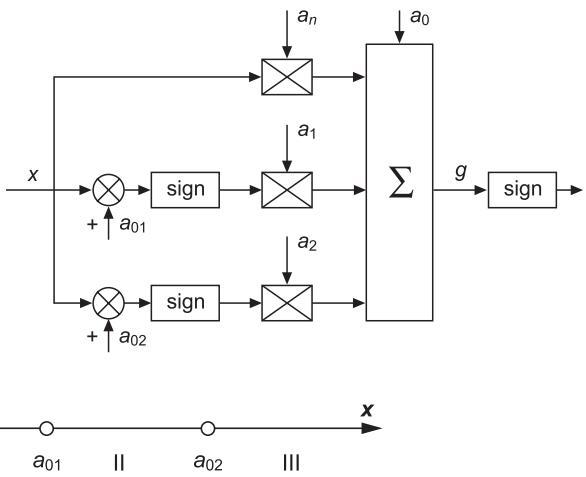


Fig. 3.1. Two-layer neural network with a cross connection, one-dimensional variant

Fig. 3.2. To the principle of cross connection in multilayer neural networks

In the process of analysis of a multilayer neural network, one must know the maximum number of regions into which the feature space of dimensionality N can be divided by H_1 hyperplanes. The maximum number of regions Ψ_{NH_1} is determined according to the following recurrent Eq. (3.1):

$$\Psi_{NH_1} = \Psi_{NH_1-1} + \Psi_{N-1,H_1-1} \tag{3.1}$$

or in the non-recurrent form

$$\Psi_{NH_1} = C^N_{H_1-1} + 2\sum_{i=0}^{N-1} C^i_{H_1-1}$$

It is implied that $C^s_t = 0$ if $t < s$. The following expressions can be derived from (3.1):

$$\Psi_{NH_1} = 2^{H_1} \quad , \quad \text{if} \quad H_1 \le N \tag{3.2}$$

and

$$\Psi_{NH_1} \le 2^{H_1} \quad , \quad \text{if} \quad H_1 > N \tag{3.3}$$

3.3 Calculation of Upper and Lower Estimation of the Number of Regions

Let us consider a multidimensional variant ($i = 1, \ldots, N$) of the neural network with the structure shown in Fig. 3.3.

The number of areas formed by the division of the initial feature space by the (j – 1)-layer neural network is designated as $\Psi_{N,[j-1]}$, where [j–1] conditionally indicates the equivalent number of hyperplanes realized by the multilayer neural network with full cross connections and with the (j–1)-th layer. Such a network consists of

$$L_{j-1} = \sum_{i=1}^{j-1} H_i$$

neurons, where H_i is equal to the number of neurons in the i-th neural network layer. The input channels of each h_j-th neuron of the j-th layer ($h_j = 1, \ldots, H_j$) can be divided into two sets, as it follows from the block scheme. The first set consists of the input signals of the neural network. The second one consists of the output signals of the

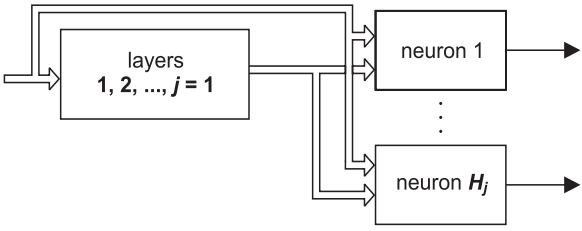


Fig. 3.3.
To the solution of the problem of optimization of the open-loop neural network structure with full sequential connections

neural network from the first, second, … (j–1)-th layers. Then the equation of the divisional surface realized by one h_j-th neuron of the j-th layer takes the form

$$a_{hj1}^T x - a_{hj}^0 - a_{hj2}^T x_{k,j-1} = 0$$

Here a_{hj1} is the vector of adjustable weights of the neural network input signals of the h_j-th neuron; a_{hj2} is the vector of adjustable weights of the input and intermediate signals of the (j–1)-layer network of the h_j-th element; a_{hj}^0 is the threshold of the h_j-th neuron; and $x_{k,j-1}$ is the vector of the neuron output signal of the (j–1)-th layer.

Therefore, relative to the initial feature space, each neuron in the j-th layer implements as much parallel hyperplanes as it is the number of variants of $x_{k,j-1}$ that are generated by the (j–1)-layer neural network. Let us assume that there exists such an adjustment procedure in which all hyperplanes generated by vector $x_{h,j-1}$ of the j-th layer belong to the initial feature space area corresponding to it. Then the recurrent equation for the calculation of the upper estimation of number of regions can be written in the form

$$\Psi_{N[j]} = \Psi_{N[j-1]} \Psi_{NH_j} \tag{3.4}$$

This follows from the fact that each of the $\Psi_{N[j-1]}$ regions selected by the (j–1)-layer sub-network is divided into Ψ_{NHj} regions, where Ψ_{NHj} is determined by the recurrent relationship (3.1).

Let us now find a non-recurrent equation. Equation (3.4) and the fact that the first layer of the neural network divides the feature space into Ψ_{NHj} areas result in

$$\Psi_{N[j]} = \prod_{i=1}^{j} \Psi_{NH_j} \tag{3.5}$$

In order to derive the estimation for the lower number of regions, let us assume that several hyperplanes out of $\Psi_{N[j-1]}$ hyperplanes can be put through any point of the initial space by the change of only a free term in the equation of hyperplane. These hyperplanes can belong to any area. The system of linear equations of relatively adjustable weights and the threshold of the h_j-th element for the estimation of their numbers has the form

$$\Psi_{N[j-1]} = \left\{ \underbrace{\begin{bmatrix} 0 & \dots & 0 & 0 & 1 & x^T \\ 0 & \dots & 0 & 1 & 1 & x^T \\ 0 & \dots & 1 & 0 & 1 & x^T \\ \cdot & \dots & \cdot & \cdot & \cdot & \cdot \\ \cdot & \dots & \cdot & \cdot & \cdot & \cdot \\ \cdot & \dots & \cdot & \cdot & \cdot & \cdot \\ 1 & \dots & 1 & 1 & 1 & x^T \end{bmatrix}}_{L_{j-1}} \times \begin{bmatrix} a_{1h_j2} \\ a_{2h_j2} \\ \vdots \\ a_{L_{j-1}h_j2} \\ 0 \\ a_{h_j} \\ a_{h_j1} \end{bmatrix} \right\} = \begin{bmatrix} q_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ q_{\Psi N[j-1]} \end{bmatrix} \tag{3.6}$$

Here q_i ($i = 1, \ldots, \Psi_{N[j-1]}$) are arbitrary given values. The number of values q_i that satisfies (3.6) is a required number of hyperplanes out of the number $\Psi_{N[j-1]}$ that can be put through any point of the initial space. This number is $(L_{s-1} + 1)$ and equals the dimensionality of vector a_{hj2} plus unit, as it follows from (3.5).

Then the following recurrent equation for the calculation of the lower estimation of number of regions is valid:

$$\Psi_{N[j]} = \Psi_{N[j-1]} - \left(L_{j-1} + 1\right) + \left(L_{j-1} + 1\right)\Psi_{NH_j}$$

Here $\Psi_{N[j-1]} - (L_{j-1} + 1)$ is the number of regions in which new hyperplanes are not put through; $(L_{j-1} + 1)\,\Psi_{NHj}$ is the number of new regions that emerge after the partition.

Finally one obtains

$$\left.\begin{aligned} \Psi_{N[j]} &= \Psi_{N[j-1]} + \left(L_{j-1} + 1\right)\left[\Psi_{NH_j} - 1\right] \\ L_j &= L_{j-1} + H_j \end{aligned}\right\} \tag{3.7}$$

Expression (3.7) is the final result. In a one-dimensional case, (3.7) obtains the form

$$\left.\begin{aligned} \Psi_{1[j]} &= \Psi_{1[j-1]} + \left(L_{j-1} + 1\right)H_j \\ L_j &= L_{j-1} + H_j \end{aligned}\right\} \tag{3.8}$$

3.4 Particular Optimization Problem

One can formulate several problems of the multilayer cross connection neural network structure optimization.

1. The number of layers and the number of neurons are given. It is required to find the neuron's distribution in the layers that maximize the number of regions Ψ formed piecewise linear divisional surface realized by the given multilayer neural network in the initial feature space.
2. The total number of neurons is given. It is required to find the number of layers and neuron's distribution in the layers that maximize the number of regions Ψ.
3. The number of regions Ψ that must be realized by the network and the number of neurons are given. It is required to find the structure that minimizes the number of neural network elements.
4. The number of regions Ψ is given. It is required to find the structure (the number of layers and neuron's distribution in the layers) that minimizes the number of network neurons. Note that the structure optimization by the number of regions represents a particular optimization criterion of the neural network.

Let us consider the structure synthesis of a one-dimensional variant of the network for the aforementioned optimization problems.

1. The number of layers W and the number of neurons

$$\sum_{j=1}^{W} H_j$$

are given. Let us find the neuron's distribution in the layers that maximize the number of regions Ψ_{1W}. Formally the problem is stated in the form of relationships written with consideration to (3.5) and (3.1):

$$\left.\begin{aligned} &\sum_{j=1}^{W} H_j = H \\ \Psi_{1[W]}^{\text{opt}} = \max_{H_1,\ldots,H_W} &\prod_{j=1}^{W}\left(H_j+1\right) \end{aligned}\right\}$$

The Lagrange method of multipliers gives the solution in the form of the system of equations

$$\left.\begin{aligned} &\prod_{\substack{j=1\\ j\neq i}}^{W}\left(H_j+1\right)+\lambda=0, \quad i=1,\ldots,W \\ &\sum_{j=1}^{W} H_j - H = 0 \end{aligned}\right\} \tag{3.9}$$

The solution of the system (3.9) is

$$\left.\begin{aligned} H_1 &= \frac{1}{W}H \quad (j=1,\ldots,W) \\ \lambda &= -\left(\frac{H}{W}+1\right)^{W-1} \end{aligned}\right\} \tag{3.10}$$

It follows from (3.10) and (3.5):

$$\Psi_{1[W]}^{\text{opt}} = -\left(\frac{H}{W}+1\right)^{W} \tag{3.11}$$

i.e., under the given number of layers the given neurons must be uniformly distributed among layers. In relationship with (3.10), the question about integrality of H_j ($j = 1, \ldots, W$) arises. If H cannot be divided by W integrally, then it follows from (3.5) and (3.10) that the remaining elements must also be distributed uniformly among layers in an arbitrary way.

In this sense, Eq. (3.11) is the upper estimation of $\Psi_{1[W]}^{\text{opt}}$ that becomes a precise upper estimation when $H = KW$, where K is an integer number.

2. The total number of layers W is not given beforehand. The number of neurons H in the network is limited. Let us find an optimum by the upper estimation structure. This can be expressed in the following form:

$$\left.\begin{aligned} \Psi_1^{\text{opt}} &= \max \Psi_{1[W]}^{\text{opt}} \\ \sum_{j=1}^{W} H_i &= H \end{aligned}\right\}$$

From the evident inequality followed from (3.11), one gets

$$\left(\frac{H}{W}+1\right)^W < \left(\frac{H}{W+1}+1\right)^W$$

and consequently the number of regions is a monotonic increasing quantity with the increase of the number of layers. Consequently, the H-layer neural network with one neuron in each layer is optimal. It follows from (3.11) that for this network,

$$\Psi_1^{\text{opt}} = \left(\frac{H}{H}+1\right)^H = 2^H$$

is the precise upper estimation.

3. The number of layers W and the number of neurons are given. Let us find the structure optimum by the lower estimation. In order to do that, let us represent (3.8) in the non-recurrent form:

$$\Psi_{1[W]} = 1 + \sum_{j=1}^{W} H_j + \frac{1}{2}\left(\sum_{j=1}^{W} H_j\right)^2 - \frac{1}{2}\sum_{j=1}^{W} H_j^2 \tag{3.12}$$

According to the Lagrange method of multipliers, constrained extremum (3.12), under condition

$$\sum_{i=1}^{W} H_i = H$$

is achieved in the case when H_i $(i = 1, \ldots, W)$ is a solution of the system

$$\left.\begin{aligned} H_i(\lambda - 1) + H + 1 &= 0 \\ \sum_{i=1}^{W} H_i - H &= 0 \end{aligned}\right\}$$

Then $H_1 = H_2 = \ldots = H_W = (1/W)H$ and under given W,

$$\Psi_{1[W]}^{\text{opt}} = 1 + H + \frac{H^2}{2} - \frac{H^2}{2W} \tag{3.13}$$

It follows from (3.13) that $\Psi^{opt}_{1[W]}$ is a monotonic increasing quantity when $W \to \infty$ and it becomes precise when $H = KW$, where K is an integer number. Consequently, $\Psi^{opt}_{1[W]} = 1 + (H + H^2)/2$ for the H-layer neural network with one neuron in the layer. Hence, in the one-dimensional case ($N = 1$), the structure's optimum by the lower and upper estimations coincide.

4. Similar to the one-dimensional case, in a multidimensional neural network variant optimum by the upper estimation, on the basis of (3.5), one can write

$$\Psi^{opt}_{N[W]} = \max_{W<H} \left\{ \max_{H_1,\ldots,H_N} \left\{ \prod_{j=1}^{W} \Psi_{NH_j} \right\} \right\} \quad \sum_{j=1}^{W} H_j = H \tag{3.14}$$

It follows from (3.14), (3.2) and (3.3) that a whole class of structures satisfies the optimal conditions (3.14). Namely, for all structures with $H_j \le N$ ($j = 1, \ldots, W$),

$$\sum_{j=1}^{W} H_j = H \tag{3.15}$$

For these structures,

$$\Psi^{opt}_{N} = 2^{\sum_{j=1}^{W} H_j} = 2^H \tag{3.16}$$

For the structures with $H_j > N$, for each $j = 1, \ldots, W$: $\Psi^{opt}_{N[W]} < 2^H$.

3.5 Structural Optimization by Some Main Topological Characteristics

A natural desire to decrease the total number of neural network inputs emerges when technological implementation of the network is performed. This is due to the fact that the number of inputs is the number of technologically complex realized multiplier units. The number of inputs at the selected structure adjustment stage is equal to the tuned coefficient space dimensionality. Therefore, the decrease of the number of neuron inputs of the multilayer neural network simplifies both implementation and adjustment.

The total number of neuron inputs in the i-th layer of the neural network with full cross connections is equal:

$$\gamma_j = \left(L_j + N\right) H_{i_j} \quad , \quad i_j = 1, \ldots, W$$

Consequently, the expression for the total number of neuron inputs in the W-layer neural network is

$$\gamma_W = \sum_{j=1}^{W} \gamma_j = \sum_{j=1}^{W} \left[\sum_{j=1}^{i} H_i + N \right] H_j = N \sum_{j=1}^{W} H_j + \frac{1}{2} H^2 - \frac{1}{2} \sum_{j=1}^{W} H_j \tag{3.17}$$

The problem of synthesis of the multilayer neural network with full cross connections and optimal by the upper or lower estimation for the number of regions under the limitations upon the number γ of network inputs is formulated on the basis of (3.17) in the following way:

$$\left.\begin{aligned} \Psi^{*}_{N[W]} &= \max_{W} \max_{H_1,\ldots,H_W} \Psi_{N[W]} \\ \gamma &\geq N\sum_{j=1}^{W} H_j + \frac{1}{2}\left[\sum_{j=1}^{W} H_j\right]^2 - \frac{1}{2}\sum_{j=1}^{W} H_j^2 \end{aligned}\right\} \tag{3.18}$$

Index * indicates an extremum value. Taking into account (3.17), the inverse problem, i.e., the problem of synthesis of structure for a multilayer neural network with full cross connections and minimum total number of inputs under the limitation on the number of regions Ψ, has the following form:

$$\gamma_W = \max_{W} \max_{H_1,\ldots,H_W} \left\{ \left[N\sum_{j=1}^{W} H_j + \frac{1}{2}\left[\sum_{j=1}^{W} H_j\right]^2 - \frac{1}{2}\sum_{j=1}^{W} H_j^2 \right] \right\} \tag{3.19}$$
$$\Psi_{N[W]} \geq \Psi$$

The number of regions $\Psi_{N[W]}$ in (3.18) and (3.19) is determined either by (3.5) or by (3.7) depending on the form of estimation.

The inverse problem statement for synthesis is less practical then the direct one because the limitation on the number of input channels in the neural network is more physically valid then the limitation on the number of regions.

Example 1. Let us show that the structure of the multilayer neural network optimum by the upper estimation of the number of regions with a limitation on the number of elements in the one-dimensional case will be optimal by the upper estimation with a limitation on the total number of inputs. According to the Lagrange method of multipliers and (3.18) with $N = 1$, optimal H_1 and H_W are the solutions of the following system of equations:

$$\left.\begin{aligned} \prod_{\substack{i=1 \\ j=j_1}}^{W} (H_j + 1) + \lambda\left[1 + \sum_{\substack{j=1 \\ j\neq B_1}}^{W} H_j\right] &= 0, \;\; j_1 = 1,\ldots,W \\ \sum_{j=1}^{W} H_j + \frac{1}{2}\left[\sum_{j=1}^{W} H_j\right]^2 - \frac{1}{2}\sum_{j=1}^{W} H_j^2 - \gamma &= 0 \end{aligned}\right\} \tag{3.20}$$

Here λ is the Langrangian multiplier.

The solution of (3.20) is

$$\left.\begin{aligned} H_1 &= 1, \quad j = 1,\ldots,W \\ \lambda &= -\frac{2^{W-1}}{W} \end{aligned}\right\} \tag{3.21}$$

Here W is the number of layers. It represents an integer part of the positive root of the equation $W^2 + W - \gamma/2 = 0$.

Equation (3.21) proves the initial statement.

Example 2. Let us consider the synthesis of the neural network taking into account (3.19), where $\Psi_{N[W]}$ is determined by the expression (3.5). Note that transference of a neuron from the j-th layer into the $(j–j_2)$-th layer causes the decrease of the number of neuron inputs for the neural network with full cross connections by

$$\Delta\gamma^{-} = \sum_{j_1=j-1}^{j-j_2} H_{j_1} \tag{3.22}$$

and the total increase of the input numbers of the rest of the neural network elements will amount to

$$\Delta\gamma^{+} = \sum_{j_1=j}^{j-j_2+1} H_{j_1} \tag{3.23}$$

It follows from (3.22) and (3.23) that the transference of a neuron from the j-th layer into the $(j–j_2)$-th layer causes the decrease of multilayer neural network input channels if

$$\sum_{j_1=j-1}^{j-j_2} H_{j_1} > \sum_{j_1=j}^{j-j_2+1} H_{j_1} \quad \text{or} \quad H_{j-j_2} > H_j$$

It follows from (3.22) and (3.23) that two structures with the total number of network neurons $H = [\log_2 \Psi]$ (square brackets mean rounding up) satisfy the optimization condition (3.19):

$$H_1 = \Delta;\ H_j = N,\ j = 2,\ldots,W$$

$$H_1 = N;\ H_W = \Delta,\ j = 1,\ldots,W-1$$

Here Δ is a remainder of division of H by N;

$$W = \frac{H-\Delta}{N} + \delta \ , \quad \text{where} \quad \delta = \begin{cases} 0, & \Delta = 0 \\ 1, & \Delta \neq 0 \end{cases}$$

Both structures correspond to the equal number of inputs determined by (3.17).

Example 3. Let us consider the optimization of the neural network structure that is optimal by the length of connections.

Let us assign some weight $U_{j_1 j}$ to each connection from the j_1-th to the j-th layer. Let us also designate the connection length of the input vector with the j-th layer as U_{0j}.

Then the total length of connections between neurons in the j-th layer is described by the expression

$$V_1 = H_1\left(U_{0j}N + \sum_{j_1=1}^{j-1} H_{j_1}U_{j_1 j}\right)$$

The total connection length in the W-layer neural network is evidently equal to

$$V_W = \sum_{j=1}^{W} H_j\left(U_{0j}N + \sum_{j_1=1}^{j-1} H_{j_1}U_{j_1 j}\right) \tag{3.24}$$

Similarly to (3.19), one can write

$$\left.\begin{aligned} \Psi^*_{N[W]} &= \max_{W}\ \max_{H_1,\ldots,H_W}\ \Psi_{N[W]} \\ V &\geq \sum_{j=1}^{W} H_j\left(U_{0j}N + \sum_{j_1=1}^{j-1} H_{j_1}U_{j_1 j}\right) \end{aligned}\right\} \tag{3.25}$$

$$\left.\begin{aligned} V^*_W = \min_{W}\ \min_{H_1,\ldots,H_W}\ \sum_{j=1}^{W} H_j\left(U_{0j}N + \sum_{j_1=1}^{j-1} H_{j_1}U_{j_1 j}\right) \\ \Psi_{N[W]} \geq \Psi \end{aligned}\right\} \tag{3.26}$$

The number of regions Ψ in (3.25) and (3.26) is determined by the expressions (3.5) or (3.7), depending on the form of estimation. Note that at $U_{j_1 j} = 1$ ($j_1 = 0, 1, \ldots, W-1$; $j = 1, \ldots, W$) the expression (3.29) coincides with (3.17), and expressions (3.25) and (3.26) coincide respectively with (3.18) and (3.19).

Example 4. Let us consider now the most general limitations on the neural network structure. They include all the aforementioned limitations as particular cases. The cost of one neuron will be β_H, the cost of one input will be β_α, and the cost of one connection unit will be β_u. Then according to (3.17) and (3.24), the total cost takes the form

$$\begin{aligned} S_W = \beta_H \sum_{j_1=1}^{W} H_{j_1} + \beta_\alpha &\left[N\sum_{j_1=1}^{W} H_{j_1} + \frac{1}{2}\left(\sum_{j_1=1}^{W} H_{j_1}\right)^2 - \frac{1}{2}\sum_{j_1=1}^{W} H_{j_1}^2\right] \\ + \beta_u &\left[\sum_{j_1=1}^{W} H_j U_{0j} N + \sum_{j_1=1}^{W} H_j U_{j_1 j}\right] \end{aligned} \tag{3.27}$$

Let us formulate the problems of neural network synthesis with a limitation on the cost S_W in a similar form to (3.18) and (3.19):

$$\left.\begin{aligned} \Psi^{*}_{N[W]} = \max_{W} \max_{H_1,\ldots,H_W} \Psi_{N[W]} \\ S \geq S_W \end{aligned}\right\} \tag{3.28}$$

$$\left.\begin{aligned} S^{*}_{W} = \min_{W} \min_{H_1,\ldots,H_W} S_W \\ \Psi_{N[W]} \geq \Psi \end{aligned}\right\} \tag{3.29}$$

The cost S_W in expressions (3.28) and (3.29) is determined by (3.27), and the number of regions $\Psi_{N[W]}$ is determined by (3.5) or (3.7) depending on the form of estimation.

All the problem statements considered above for the neural network structure synthesis can be obtained by variation of cost coefficients β_H, β_α, and β_u in (3.28) and (3.29).

3.6 Optimization of a Multilayer Neural Network Structure with K_p Solutions

The elements of a multilayer neural network with full cross connections are described by expressions represented in the previous chapter. Each element realizes an assemblage of parallel divisional hyperplanes in its feature space. The maximum number of regions selected in the initial feature space by the equivalent divisional surface does not exceed K_p^H, where H is the number of network neurons. This estimation is achieved only for a multilayer neural network with full cross connections.

Let us estimate a number of regions that can emerge as a result of the partitioning of the N-dimensional feature space by H_1 groups of hyperplanes consisting of $(K_p–1)$-th hyperplane in each group. Let us designate a maximum number of regions selected by the $[H_1–1]$ group as $\Psi^{K_p}_{N[n-1]}$. Then similar to (3.3) and (3.4)

$$\Psi^{K_p}_{N[H_1]} = \Psi^{K_p}_{N[N_1-1]} + z$$

Let us estimate a z value. When placing each of the $(K_p – 1)$ hyperplanes, the number of selected regions increases by the number of regions formed on the hyperplane by lines of its intersection with other hyperplanes, i.e., by $\Psi^{K_p}_{N-1[H_1-1]}$.

Consequently,

$$z = \left[K_p - 1\right] \Psi^{K_p}_{N[H_1-1]}$$

and finally

$$\Psi^{K_1p}_{N[H_1]} = \Psi^{K_1p}_{N[H_1-1]} + \left[K_p - 1\right]^{K_p}_{N[H_1-1]} \tag{3.30}$$

with the initial conditions

$$\Psi_{N[1]}^{K_p} = K_p \quad ; \quad \Psi_{N[H_1]}^{K_p} = H_1\left[K_p - 1\right] + 1 \tag{3.31}$$

It can be shown from (3.30) and (3.31) that

$$\begin{aligned} &\Psi_{N[H_1]}^{K_p} = K_p^{H_1} \quad \text{when} \quad H_1 \le N \\ &\Psi_{N[H_1]}^{K_p} < K_p^{H_1} \quad \text{when} \quad H_1 > N \end{aligned} \tag{3.32}$$

Let us consider the h_j-th neuron with K_p solutions located in the j-th layer of the multilayer neural network with full cross connections. The input signals of the h_j-th neuron can be divided into two groups: input signal vector $x = [x_1, \ldots, x_N]$ and row-vector $y = [y_1, \ldots, y_L]$ of output and intermediate signals of the (j–1)-layer neural network. Let us assume that the (j–1)–layer neural network selects $\Psi_{N[j-1]}^{K_p}$ regions in the initial feature space.

Then $\Psi_{N[j-1]}^{K_p}$ different variants of vector y can come through the y-channels to the inputs of the h_j-th neuron. The equation for the output signal of the h_j-th neuron with K_p solutions can be written in the following form:

$$x_{kh_j} = F_p\left(A_{h_j} x + A_{h_j}^1 y\right) \quad , \quad h_j = 1, \ldots, H_j \tag{3.33}$$

where A_{h_j} and $A_{h_j}^1$ are the vectors of weighting coefficients for x and y, respectively.

Each of the H_j neurons with K_p solutions realizes geometrically (K_p–1) $\Psi_{N[j-1]}^{K_p}$ parallel hyperplanes in the neural network input signal space according to (3.33). The expression for the upper estimation of the number of regions that emerges as a result of decomposition of X space by the considered j-layer neural network has the form

$$\Psi_{N[j]}^{K_p} = \Psi_{N[j-1]}^{K_p} \Psi_{NH_j}^{K_p} \tag{3.34}$$

Here $\Psi_{NHj}^{K_p}$ is determined by (3.30) and (3.31). Taking into consideration that (3.34) is a recurrent expression and that the first neuron layer with K_p solutions decomposes the X space into $\Psi_{NHj}^{K_p}$ regions, the expression (3.34) can be rewritten as

$$\Psi_{N[j]}^{K_p} = \prod_{i=1}^{j} \Psi_{NH_i}^{K_p} \tag{3.35}$$

The expression (3.35) allows one to formulate and to solve the problem of synthesis of the neural network structure optimum by the upper estimation of the number of regions under the limitation on the total number N of neurons in the neural network. It follows from (3.35) and (3.32) that in the W-layer neural network

$$\left.\begin{aligned} &\Psi_{N[W]}^{K_p} = K_p^{\sum_{j=1}^{W} H_j} \quad \text{if} \quad H_j \le N \\ &\Psi_{N[W]}^{K_p} < K_p^{\sum_{j=1}^{W} H_j} \quad \text{if} \quad H_j > N \quad (j = 1, \ldots, W) \end{aligned}\right\} \tag{3.36}$$

Consequently, the following neural network with full cross connections will be optimal by the upper estimation of the number of regions: the number of neurons with K_p solutions in each layer of such a neural network must not exceed the dimensionality of the initial feature space.

Literature

[3-1] Lupanov OB (1958) About possibilities of synthesis of circuits with arbitrary number of elements. Proc. of Steklov Mathematical Institute 51:158–173

[3-2] Galushkin AI (1974) Synthesis of multilayer systems of pattern recognition. Moscow, Energyia

[3-3] Galushkin AI, Schmidt AB (1992) Optimization of structure of multilayer neural networks with cross connections. Neurocomputer 3,4

Chapter 4

Continual Neural Networks

A large number of parameters characterizing the input signal must be taken into consideration while developing multilayer neural network structures. An example is a design of a pattern recognition system with the requirement of maximum probability of correct recognition [4-1]. It is suggested to take into account continual properties of multilayer neural network characteristics in its mathematical modeling and in its technological implementation.

4.1 Neurons with Continuum Input Features

The transfer to the continuum features becomes important when the dimensionality N of some feature vector x_i becomes large (several hundreds or thousands). The feature vector in this case $\{\mathbf{x}_i,\ i=1,\ldots,N\}$ is replaced by the function of indiscrete argument $\{x(i),\ i\in I\}$, and the weighting vector $\{\mathbf{a}_m, m=1,\ldots,M\}$ is replaced by the weighting function $\{a(m), m\in M\}$. The neuron model with a continuum of features, similar to the classic discrete case [4-2], is determined by the expression

$$y=\operatorname{sign}\left(\int_M a(m)x(m)\,\mathrm{d}m+a_0\right) \tag{4.1}$$

where y is a neuron output signal; $x(m)$ is a neuron input signal; and a_0 is a threshold value.

The transfer to the feature continuum at the input of the first neural network layer often excludes the requirement to quantize the input signal (for example, electric signal or pattern, etc.). The method of technological implementation of the neuron weighting function is selected depending on the concrete physical type of the input signal. For example, when the input signal is an electric one changing in time, the weighting function must also be generated as an electric signal. In the case of optical patterns, the weighting function can be realized on the photomask. For the discrete set of neurons with continuum feature space,

$$y_{m_2}=\operatorname{sign}\left(\int_{M_1} x(m_1,k)a_{m2}(k)\,\mathrm{d}m_1+a_0,m_2\right) \tag{4.2}$$

The neural network structure according to (4.2) is shown in Fig. 4.1.

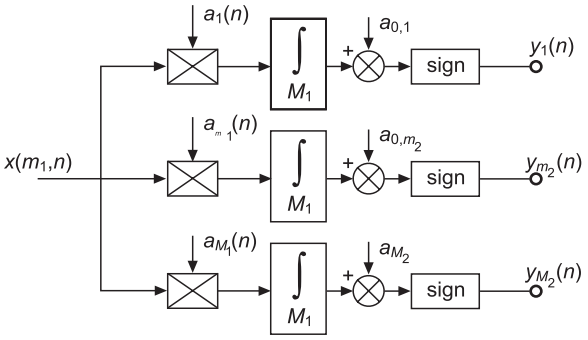


Fig. 4.1.
Neural network layer with continuum input features

4.2
Continuum of Neurons in the Layer

The transfer to the continuum space of the layer output signals can be interpreted as a continuum of neurons in the layer. Then the output signal is realized not in the form of a finite-dimensional vector, for example consisting of + 1 and –1, as it is shown in Fig. 4.1, but in the form of a $y(m_2, n)$ signal having values + 1 and –1 in the interval of variance of indiscrete argument m_2. Consequently, taking into account (4.2), the output signal is an infinite-dimensional vector with components

$$y(m_2,n)=\operatorname{sign}\left(\int_{M_1} x(m_1,k)a(m_2,m_1)\mathrm{d}m_1+a_0(m_2)\right) \tag{4.3}$$

The expression (4.3) forms the basis for physical implementation of open-loop systems of the type considered.

4.3
Continuum Neurons in the Layer and Discrete Feature Set

In the particular case of discrete feature set and continuum neurons in the layer under the transfer from indiscrete variable m_1 to the discrete one, the expression (4.3) takes the form

$$y(m_2,n)=\operatorname{sign}\left(\sum_{m_1}^{M_1} x_{m_1}(n)a_{m_1}(m_2)+a_0(m_2)\right) \tag{4.4}$$

This expression is the basis for the implementation of the neural network whose particular case is represented in Fig 4.2.

The neuron's layer output signal is an electric signal with form features at the n-th period. Electric signals $a_{m_1}(m_2)$ and $a_0(m_2)$ are generated inside the system at each n-th step. As distinct from the system of signal form recognition, their period is determined a priori. The electric signal of two values (1, –1) in the interval $(0, M_2)$ emerges at the layer output (Fig. 4.3).

Such a layer model with a continuum of neurons is adequate for the neurophysiological neuron model under the pulse-frequency modulation of the layer output signal.

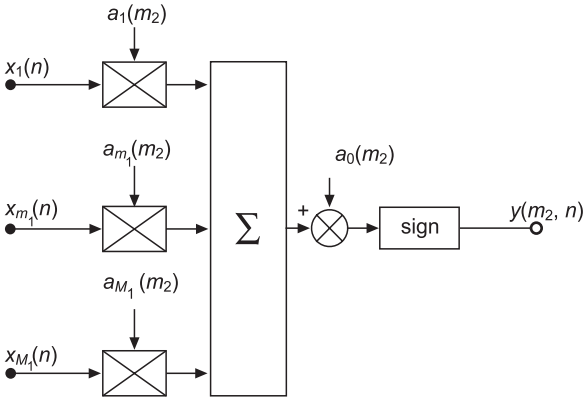


Fig. 4.2.
Discrete feature set. Continuum of neurons in the layer

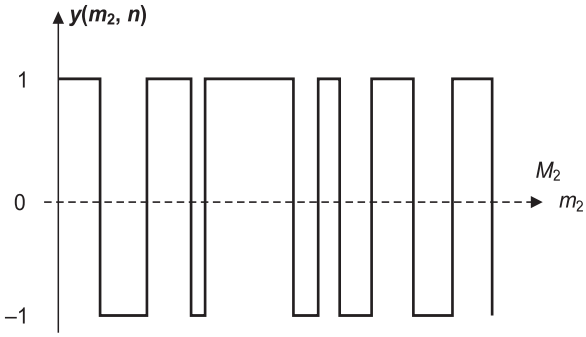


Fig. 4.3.
Form of the neuron layer output signal

4.4
Classification of Continuum Neuron Layer Models

The systems considered below serve as the recognition of one-channel electric signals of the "bump" type, i.e., the signal with lock-in bump onset and periodic signal with lock-in at each period (Fig. 4.4).

4.4.1
Discrete Set of Neurons

A basis of the neural network implementation in this case is the expression (4.2) in which $y_{m_2}(n)$ has two values $(1, -1)$; $x(m_1, n)$ and $a_{m_2}(n)$ are electric signals at the n-th period; a_{0m_2} is a constant coefficient.

4.4.2
One-Dimensional and Two-Dimensional m_2 Feature Space

A basis of the neural network implementation in this case is the expression (4.3). The difficulty for implementation is a multiplier unit for the multiplication of a continuous function of one variable by a continuous function of two variables, one of which coin-

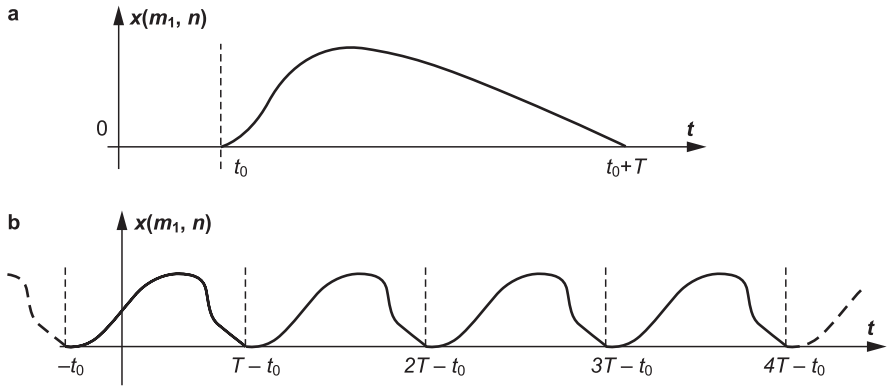


Fig. 4.4. Types of signals received by the neuron layer in the case of features one-dimensional and one-channel continuum m_1: *a* – bump signal; *b* – periodic signal

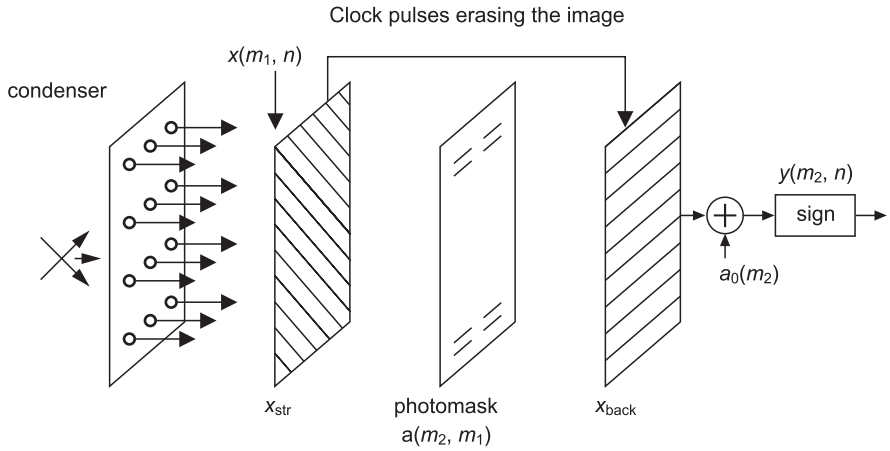


Fig. 4.5. Hypothetical variant of optical system implementation in the case of one-dimensional one-channel m_1 and two-dimensional m_2

cides with the variable of the first function. The requirement of the continuum type of the layer input space makes it necessary to realize the function $x(m_2, n)$ in the form of an electric signal. At the same time, the requirement of the continuum type of the layer output space makes it necessary to realize the function $a(m_2, m_1)$ in the form of an image. The main difficulty is the multiplication of functions $x(m_1, n)$ and $a(m_2, m_1)$. One can assume the physical existence of the optical element x_{str}, whose dimness (or brightness) changes in the real time along one of the coordinates depending on the form of applied voltage, and the form of the output voltage – depending on the dimness (or brightness) distribution along the coordinate at the integration of the intensity along another one. Such an element can be conditionally called a "spatial optical coupler". Then the implementation of this network could have the form shown in

Fig. 4.5, where the layer input electric signal would come to the vertical plate of the element x_{str} at its input, and the output signal would be read from the horizontal plate of the output element x_{back}.

Consequently, it is difficult to consider two-dimensional m_2 at the existent level of technological development because it requires the physical implementation of three-dimensional function $a(m_2, m_1)$.

4.4.3 Continuum of Features – One-Dimensional m_1 for Several Channels

This case has an important practical meaning because it is often necessary to recognize a multi-channel signal, for example, at the recognition of EEG when the simultaneous analysis of several disposals is performed, Fig 4.6.

One can use the expression (4.2) for the neural network implementation in this case. The neuron structure is shown in Fig. 4.7, and the neuron layer represents a parallel neuron connection with equal inputs (Fig. 4.8).

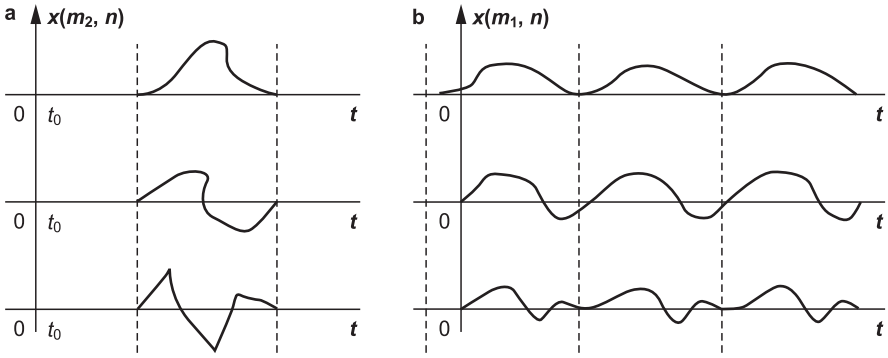


Fig. 4.6. Types of signals received by the layer of neurons in the case of a feature continuum and one-dimensional multi-channel m_1: *a* – bump signal; *b* – periodic signal

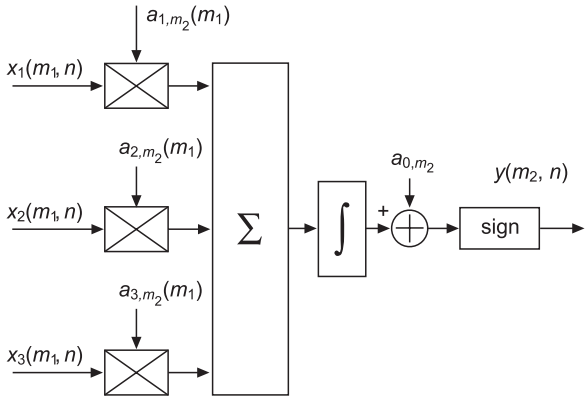


Fig. 4.7. Block-scheme of physical implementation of the neuron layer with a feature continuum and one-dimensional and multi-channel m_1

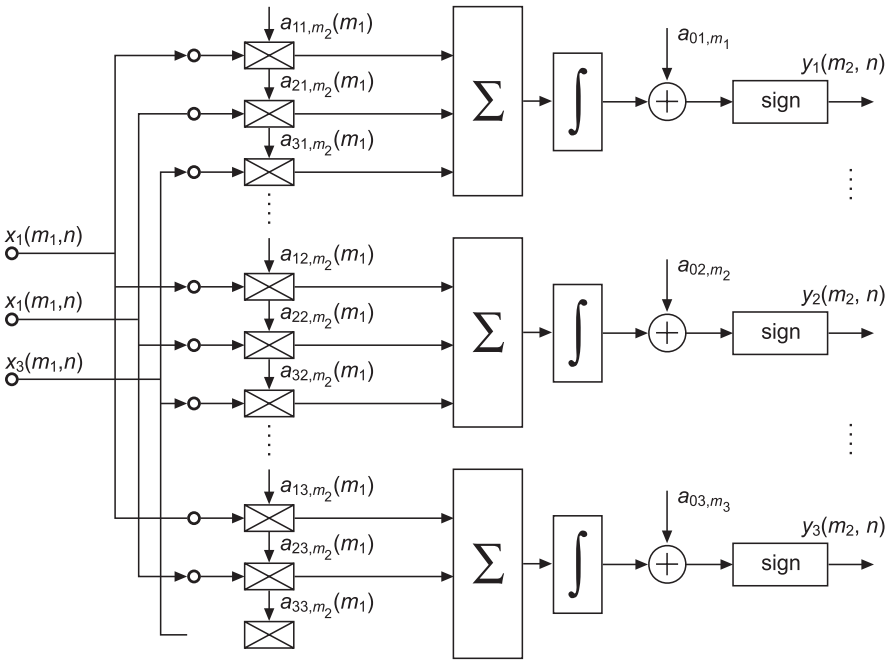


Fig. 4.8. Structure of a multi-channel neuron layer with discrete set of neurons in the case of a feature continuum and one-dimensional feature space

4.4.4
Feature Continuum – Two-Dimensional m_1

The problem of image recognition is adequate for the variant of a discrete set of neurons in the layer and two-dimensional m_1. Taking into account (4.2), the neuron layer structure in the case of a feature continuum and two-dimensional m_1 can be represented in the form shown in Fig. 4.9. The system of the image's multiplication can be realized on the basis of optical fibers or with the help of holographic methods or a system of mirrors.

4.4.5
Neuron Layer with a Continuum of Output Values

In this case, the output space remains at the bottom to be a space of table features varying in some interval of indiscrete set of values determined by activation function, i.e., by nonlinear transformation at the neuron output. This also concerns systems (4.2), Figs. 4.1, 4.9.

For systems (4.4), the output electric signal has the form of a function of one variable taking two values (1, –1) in some interval of its variance. The transfer to the neurons with a continuum of solutions results in the output electric signal varying indiscreetly by its amplitude.

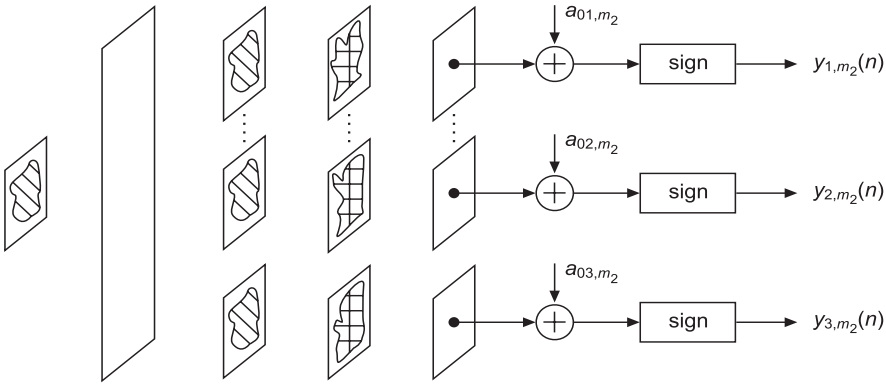


Fig. 4.9. Block-scheme of a physical implementation of a neuron layer with a feature continuum, two-dimensional m_1 and discrete set of elements in the layer

A multilayer neural network for the solution of different problems concerning vectors, signals and pattern processing can be designed as different combinations of the aforementioned continuum systems. The following problems are perspective in this field:

- Search for physical implementation methods for different types of continuum models of multilayer neural networks;
- Development of structure synthesis methods for continuum models of multilayer neural networks according to the different criteria;
- Development of other a priori given structures of continuum models of a multilayer neural network (cross connections and backward connections systems, etc.);
- Use of a system's "human-machine" for the continuum multilayer network structure synthesis;
- Introduction of new continuum properties of multilayer neural networks (continuum number of layers, etc.)

The transfer to the continuum feature space and continuum set of neurons in the layer becomes important when the number of features becomes large (several hundreds or thousands).

The problem of transfer to the continuum number of layers is not so significant. The preliminary analysis of this problem shows principle mathematical difficulties for the implementation of such a transfer. This follows from the expression for the output signal of the three-layer system:

$$y_3 = \operatorname{sign} \sum_{m_3}^{M_3} a_{m_3} \operatorname{sign} \sum_{m_2}^{M_2} a_{m_2} \operatorname{sign} \sum_{m_1}^{M_1} a_{m_1} x_{m_1}$$

The solution of the problem for the transfer to the continuum of the number of layers is complicated due to the nonlinear transducers at the output of each layer and due to the difficulty of selecting a method of physical implementation of the open-loop system adequate for such a model.

Literature

[4-1] Galushkin AI (1974) Synthesis of multilayer systems of pattern recognition. Moscow, Energiya
[4-2] Rosenblatt F (1962) Principles of neurodynamics. Spartan Books, Washington
[4-3] Galushkin AI (1977) Continuum models of multilayer systems for pattern recognition. Automation and computer technology, Riga, No. 2
[4-4] Galushkin AI (1992) Continuum neural networks. Neurocomputer 2

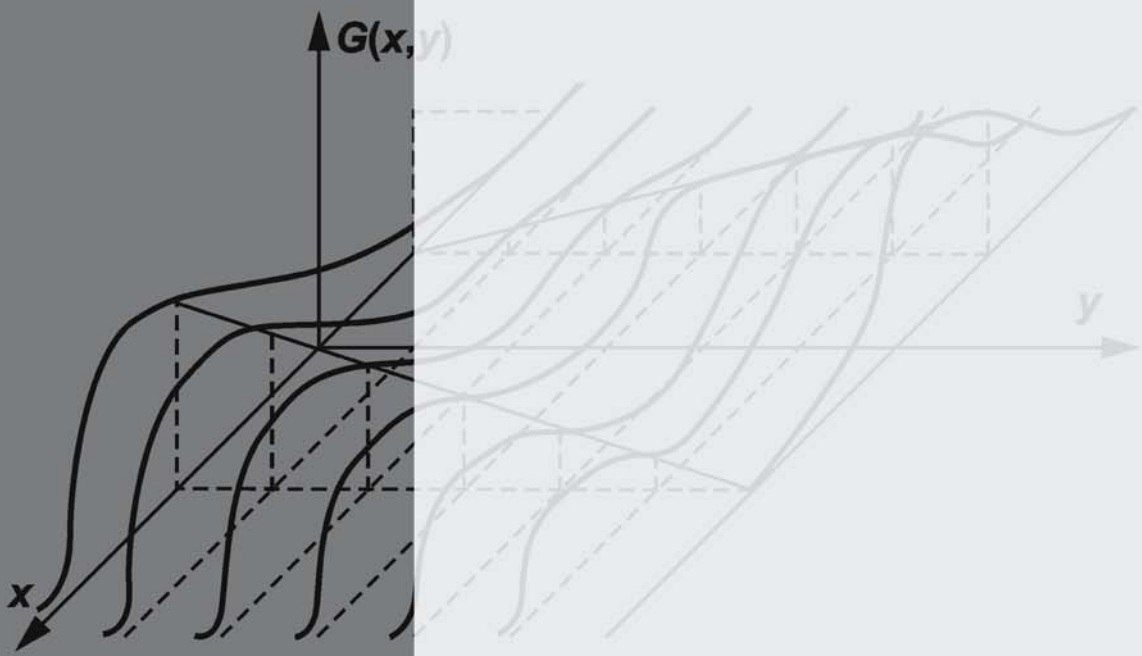
Part II Optimal Models of Neural Networks

Chapter 5 Investigation of Neural Network Input Signal Characteristics

Chapter 6 Design of Neural Network Optimal Models

Chapter 7 Analysis of the Open-Loop Neural Networks

Chapter 8 Development of Multivariable Function Extremum Search Algorithms



Chapter 5

Investigation of Neural Network Input Signal Characteristics

5.1 Problem Statement

A neural network can be represented in the form of an equivalent system that adapts to external conditions. A general block-scheme of such a system is shown in Fig. 5.1, where $\mathbf{x}(n)$ is a multidimensional stochastic process having the form of pattern sequence at the neural network input; n is a discrete argument.

Signal $\boldsymbol{\varepsilon}(n)$ is determined by supervisor instructions belonging to the current pattern at the neural network input to a particular class. Each class includes some set of patterns possessing a common property. A multidimensional output signal of the recognition system $\mathbf{y}(n)$ is generated in the form of neural network data belonging to the current pattern of a particular area in the solution space. The three spaces considered, X, E, Y, are respectively spaces of patterns, supervisor instructions and neural network output signals. The unit of neural network parameter adjustment determines vector $\mathbf{a}(n)$ of adjustable coefficients and information about $\mathbf{y}(n)$ transformation structure, i.e., dependence of the neural network output signal on the input one; $\mathbf{g}(n)$ is a vector of intermediate neural network signals.

The input signal of the neural network is $[\mathbf{x}(n), \boldsymbol{\varepsilon}(n)]$. One of its characteristics is the number of $\boldsymbol{\varepsilon}(n)$-signal level gradations determined by the number of pattern classes. Signal $\mathbf{x}(n)$ of dimensionality N can be in general discrete or indiscrete by its amplitude. If $\boldsymbol{\varepsilon}(n)$ is a one-dimensional signal with its level discretized into two or K gradations, then respectively 2 or K pattern classes are considered. If vector $\boldsymbol{\varepsilon}(n)$ has a dimensionality N^* and the number of amplitude gradations for each of its components amounts to K_0 then the number of classes is

$$K = (K_0)^{N^*}$$

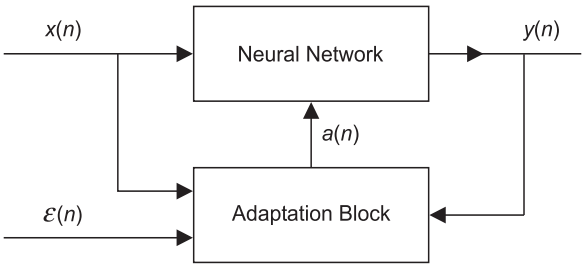


Fig. 5.1. Block-scheme of the neural network

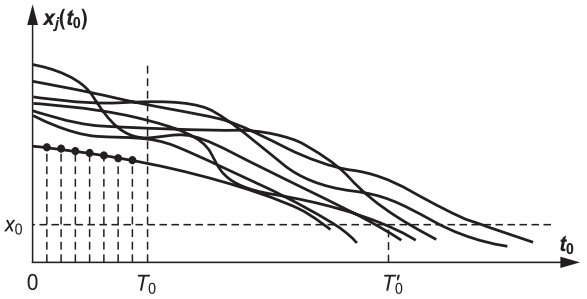


Fig. 5.2.
The formation of a feature space and supervisor instructions in the problem of device reliability forecasting

A class continuum is considered when signal $\boldsymbol{\varepsilon}(n)$ is indiscrete. Then the neural network adjustment can be considered as the problem of system estimation of some indiscrete parameter $\boldsymbol{\varepsilon}$ for the distribution $f(\mathbf{x},\boldsymbol{\varepsilon})$ of some stochastic process.

A particular problem of adjustment (learning) can be illustrated in the following example. Let us consider the formation of the neural network input signal in the case of a feature continuum in the problem of reliability forecasting of some device.

In Fig. 5.2, $x_j(t_0)$ are the curves for the time changes of some device parameter that serves as an indicator of the device reliability in the test; j is the number of the testing device; x_0 is the parameter's acceptable value.

The point of the curve $x_j(t_0)$ intersection with the level x_0 determines the device non-failure operating time. Vector $x_j(n)$ corresponding to each curve is the vector obtained by the curve discretization in time in the interval $[0, T_0]$, where T_0 is the time of the device testing. Components of the vector correspond to the ordinates $x_j(t_0)$ in the points of discretization. This procedure allows one to form the feature space in this problem.

The supervisor instruction can be formed in the following way. The device operation life time T_0' is given a priori. Vectors $x_j(n)$ determined by the points of intersection of the curves $x_j(t_0)$ and x_0 that lie before T_0' belong to the first (failure) class of devices, and after T_0' – to the second (fitting) class. Respectively, two amplitude gradations of the signal $\varepsilon(n)$ are introduced $(1, -1)$. The t_0 axis can be partitioned into K intervals with an indication of the device type, and the signal $\varepsilon(n)$ will have K amplitude gradations (for example, $\varepsilon = 1, \ldots, K$). Each vector $x_j(n)$ will have its own value ε.

In the extreme case, when the t_0 axis is not partitioned, the supervisor instruction for the system of the device's non-failure operating time has indiscrete distribution.

The following characteristics can be the object of analysis in each particular case: joint distribution $f(\mathbf{x},\boldsymbol{\varepsilon})$, conditional distribution $f'(\mathbf{x}/\boldsymbol{\varepsilon})$ of pattern $\mathbf{x}$ assemblage under a given instruction $\boldsymbol{\varepsilon}$ about its belonging to the k-th class, distribution of signals $f(\mathbf{x})$ and $f_\varepsilon(\boldsymbol{\varepsilon})$ and their different moments of distribution, etc.

The introduction of such a notion as teacher (supervisor) qualification allows one to develop a unified approach to the problems of learning and self-learning. Joint distribution of input patterns and instruction signal of patterns belonging to some class is represented on the basis of a unified approach.

5.2 Joint Probability Distribution of the Input Signal for Two Pattern Classes

In the neural network learning process, learning sample patterns belonging to a particular class is known with probability 1, i.e., the teacher (supervisor) gives the precise instruction concerning the learning sample. In the self-learning process, the signals in the learning sample are not accompanied by the instruction about their belonging, and the probability is less than 1. In the simplest case of two-mode distribution, this probability is 0.5. Let us designate as a, the probability of the supervisor instruction for the pattern to belong to some class.

It is important to analyze intermediate modes of the transfer from the problem of learning to the problem of self-learning and vice versa in the algorithm block-scheme represented in Fig. 5.3.

The transfer must be performed by the change of the learning sample member probability of belonging to a particular class in the range from $a = 1$ to $a = 0.5$ (and vice versa).

The following reasons make it necessary to analyze such intermediate modes:

1. Development of a unified approach to the analysis and synthesis of learning and self-learning modes for pattern recognition systems;
2. Solution of some practical problems. One such problem is the learning procedure with the teacher of incomplete qualification.

The expression for the joint distribution $f(\mathbf{x},\boldsymbol{\varepsilon})$ of signals $\mathbf{x}(n)$ and $\boldsymbol{\varepsilon}(n)$ is

$$f(\mathbf{x},\varepsilon)=\begin{cases} p_1(1-a)f_1(\mathbf{x})+p_2af_2(\mathbf{x}) \ , & \varepsilon=1, \\ p_1af_1(\mathbf{x})+p_2(1-a)f_2(\mathbf{x}) \ , & \varepsilon=-1, \end{cases} \tag{5.1}$$

where p_1 and p_2 are the a priori probabilities of the emergence of the first and second classes; $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are distributions of signals $\mathbf{x}_1(n)$ and $\mathbf{x}_2(n)$ representing patterns of the first and second classes.

Distribution (5.1) is a discrete-continuous one due to the discrete form of $\boldsymbol{\varepsilon}(n)$. However, it can be written in the indiscrete form using the Kronecker delta function. The discrete representation is taken into account below by means of the replacement of the integration operations with summation operations.

The level of teacher qualification b is introduced in the following way [5-1, 5-2]:

$$b = 2a - 1 \tag{5.2}$$

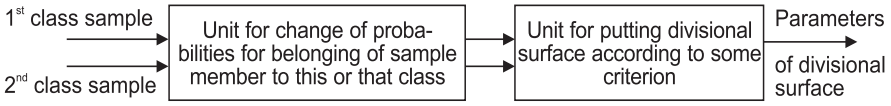

1st class sample
2nd class sample
Unit for change of probabilities for belonging of sample member to this or that class
Unit for putting divisional surface according to some criterion
Parameters
of divisional surface

Fig. 5.3. Structure of mathematical model for the unified approach to the problems of learning and self-learning in the pattern recognition systems

Consequently, $b = 1$ when $a = 1$, and the teacher qualification is complete; $b = 0$ when $a = 0.5$, and the teacher qualification is zero.

Taking into account (5.2) and (5.1), one obtains the following:

$$f(\mathbf{x},\varepsilon)=\begin{cases}\dfrac{p_1(1-b)}{2}f_1(\mathbf{x})+\dfrac{p_2(1+b)}{2}f_2(\mathbf{x}) \ , & \varepsilon=-1,\\ \dfrac{p_1(1+b)}{2}f_1(\mathbf{x})+\dfrac{p_2(1-b)}{2}f_2(\mathbf{x}) \ , & \varepsilon=1,\end{cases} \tag{5.3}$$

Figure 5.3 represents a structure of a mathematical model for the generation of a neural network input in the considered case. Expressions (5.3) and (5.2) relate to the equal teacher qualification level for both sample classes. Equation (5.3) at $b = 1$ gives the following joint distribution law for the input signal in the neural network learning mode:

$$f(\mathbf{x},\varepsilon)=\begin{cases}p_2f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon=1\\ p_1f_1(\mathbf{x}) \ , & \text{if} \quad \varepsilon=-1\end{cases} \tag{5.4}$$

The joint distribution law for the input signal in the self-learning mode at $b = 0$ has the form

$$f(\mathbf{x},\varepsilon)=\begin{cases}\dfrac{p_1}{2}f_1(\mathbf{x})+\dfrac{p_2}{2}f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon=1\\ \dfrac{p_1}{2}f_1(\mathbf{x})+\dfrac{p_2}{2}f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon=-1\end{cases}$$

Here the signal of the supervisor instruction $\varepsilon(n)$ gives no information about the pattern belonging to any class because the conditional probabilities $f'(\mathbf{x}/\varepsilon = 1)$ and $f'(\mathbf{x}/\varepsilon = -1)$ are equal.

Expression (5.3) at $b = -1$ gives

$$f(\mathbf{x},\varepsilon)=\begin{cases}p_2f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon=-1\\ p_1f_1(\mathbf{x}) \ , & \text{if} \quad \varepsilon=1\end{cases}$$

The teacher in this case performs an incorrect classification intentionally (the teacher is a "saboteur").

By definition of conditional probability

$$f'(\mathbf{x}/\varepsilon)=\frac{f(\mathbf{x},\varepsilon)}{f_\varepsilon(\varepsilon)} \tag{5.5}$$

where

$$f_\varepsilon(\varepsilon)=\int_{-\infty}^{+\infty} f(\mathbf{x},\varepsilon)\mathrm{d}\mathbf{x}$$

Integration of (5.3) gives the following supervisor instruction distribution function:

$$f_{\varepsilon}(\varepsilon)=\begin{cases}\frac{1}{2}+\frac{b}{2}(p_2-p_1)\ , & \text{if} \quad \varepsilon=1\\ \frac{1}{2}+\frac{b}{2}(p_1-p_2)\ , & \text{if} \quad \varepsilon=-1\end{cases} \tag{5.6}$$

Equations (5.6), (5.3) and (5.5) give

$$f'(\mathbf{x}/\varepsilon)=\begin{cases}\frac{p_1(1-b)f_1(\mathbf{x})+p_2(1+b)f_2(\mathbf{x})}{1-b(p_1-p_2)}\ , & \text{if} \quad \varepsilon=1\\ \frac{p_1(1+b)f_1(\mathbf{x})+p_2(1-b)f_2(\mathbf{x})}{1-b(p_1-p_2)}\ , & \text{if} \quad \varepsilon=-1\end{cases} \tag{5.7}$$

The conditional distribution law $f''(\varepsilon/\mathbf{x})$ is similarly defined as

$$f''(\mathbf{x}/\varepsilon)=\frac{f(\mathbf{x},\varepsilon)}{f(\mathbf{x})}$$

$$f(\mathbf{x})=\int_{-\infty}^{+\infty} f(\mathbf{x},\varepsilon)\mathrm{d}\varepsilon=\sum_{k'=1}^{2} p_{k'}f_{k'}(\mathbf{x})$$

where the integration with discrete argument ε is replaced by summation. One obtains after substitution and integrating

$$f''(\varepsilon/\mathbf{x})=\begin{cases}\frac{p_1\frac{1-b}{2}f_1(\mathbf{x})+p_2\frac{1+b}{2}f_2(\mathbf{x})}{p_1f_1(\mathbf{x})+p_2f_2(\mathbf{x})}\ , & \text{if} \quad \varepsilon=1\\ \frac{p_1\frac{1+b}{2}f_1(\mathbf{x})+p_2\frac{1-b}{2}f_2(\mathbf{x})}{p_1f_1(\mathbf{x})+p_2f_2(\mathbf{x})}\ , & \text{if} \quad \varepsilon=-1\end{cases} \tag{5.8}$$

It follows from (5.7) and (5.8) at $b = 0$:

$$f'(\mathbf{x}/\varepsilon)=f_x(\mathbf{x}),\ f''(\varepsilon/\mathbf{x})=f_{\varepsilon}(\varepsilon)$$

indicating statistical independence of signals $\mathbf{x}(n)$ and $\varepsilon(n)$ at the neural network input in the self-learning mode.

Let us designate the joint moment of the j-th order of multidimensional stochastic process $\mathbf{x}(n)$ as α_j:

$$\alpha_j=\int_{-\infty}^{+\infty}\dots\int_{-\infty}^{+\infty} x_{i_1}\dots x_{i_j}f_x(x_1,\dots,x_N)\mathrm{d}x_1,\dots,\mathrm{d}x_N\ , \quad i_1,\dots,i_j=1,\dots,N$$

Then the expression for the moments of distribution (5.3) has the form

$$\overline{\varepsilon^i x^j}^n = p_1 \frac{1-b}{2}\alpha_{j_1} + p_2 \frac{1+b}{2}\alpha_{j_2} + (-1)^i p_1 \frac{1+b}{2}\alpha_{j_1} + (-1)^i p_2 \frac{1-b}{2}\alpha_{j_2}$$

where a_{j1} and a_{j2} are the joint moments of the j-th order for the assemblage of patterns of the first and second classes. One obtains in the cases of even-numbered and odd-numbered i

$$\overline{\varepsilon^i x^j}^n = p_2\alpha_{j_2} + p_1\alpha_{j_1};\ \ \overline{\varepsilon^i x^j}^n = b\left(p_2\alpha_{j_2} + p_1\alpha_{j_1}\right)$$

Consequently, the teacher qualification influences the moments of distribution $f(x,\varepsilon)$ at odd-numbered i.

Unequal teacher qualification with respect to the patterns of the first and second classes. In some practical problems, the teacher qualification of the pattern recognition system is different for the patterns of the first and second classes. Let us introduce a stochastic matrix

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} a_1 & 1-a_2 \\ 1-a_1 & a_2 \end{bmatrix}$$

where a_{ij} is the probability of the supervisor instruction to consider patterns of the j-th class as patterns of the i-th class. In this case,

$$f(\mathbf{x},\varepsilon) = \begin{cases} p_1(1-a_1)f_1(\mathbf{x}) + p_2 a_2 f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon = 1 \\ p_1 a_1 f_1(\mathbf{x}) + p_2(1-a_2)f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon = -1 \end{cases}$$

or

$$f(\mathbf{x},\varepsilon) = \begin{cases} \dfrac{p_1(1-b_1)}{2} f_1(\mathbf{x}) + \dfrac{p_2(1+b_2)}{2} f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon = 1 \\ \dfrac{p_1(1+b_1)}{2} f_1(\mathbf{x}) + \dfrac{p_2(1-b_2)}{2} f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon = -1 \end{cases} \tag{5.9}$$

The analysis of different variants for the relationship between b_1 and b_2 is an object of special discussion. For example, when $b_1 = 1$ and $b_2 = 0$, the teacher qualification is equal to 1 for the first class, and is equal to 0 for the second class. In this case,

$$f(\mathbf{x},\varepsilon) = \begin{cases} \dfrac{1}{2} p_2 f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon = 1 \\ p_1 f_1(\mathbf{x}) + \dfrac{1}{2} p_2 f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon = -1 \end{cases}$$

This variant is intermediate between learning and self-learning modes.

Moments of distribution for (5.9) have the form

$$\overline{\varepsilon^i \mathbf{x}^j}^n = p_2\alpha_{j2} + p_1\alpha_{j1} \quad \text{for even-numbered } i$$

$$\overline{\varepsilon^i \mathbf{x}^j}^n = b_2 p_2 \alpha_{j2} + b_1 p_1 \alpha_{j1} \quad \text{for odd-numbered } i$$

Joint distribution law in the case of "teacher's slant about his abilities". The teacher makes some number of mistakes in the pattern recognition system learning. Let us introduce the notion "teacher's slant about his qualification" by some coefficient c. At $c > b$ we have "teacher's overestimation" of his abilities $c - b$. At $c < b$ we have "teacher's underestimation" of his abilities $b - c$. The question concerning the investigation of the influence of teacher's slant and his real qualification upon the neural network operation characteristics arises.

Similar problems can be also formulated in the sense of "sabotage" when $-1 < b < 0$.

Let us designate the supervisor instruction in the case of complete teacher overestimation as ε'. It was assumed in the previous text that $\varepsilon = \varepsilon'$. The uncertainty of the teacher in his abilities results in the fact that the classification of the appeared pattern as a pattern of the first ($\varepsilon = -1$) or second ($\varepsilon = 1$) class is performed with the probability $(1 + c)/2$. Respectively, the same patterns belonging to the second or first classes are determined with the probability $(1 - c)/2$. The joint distribution of random values ε and ε' can be represented in the form

$$f'''(\varepsilon, \varepsilon') = \begin{cases} \left.\begin{cases} p_2 \dfrac{1+c}{2}, & \text{if} \quad \varepsilon = 1 \\ p_2 \dfrac{1-c}{2}, & \text{if} \quad \varepsilon = -1 \end{cases}\right\} & \text{if} \quad \varepsilon' = 1 \\ \left.\begin{cases} p_1 \dfrac{1+c}{2}, & \text{if} \quad \varepsilon = -1 \\ p_1 \dfrac{1-c}{2}, & \text{if} \quad \varepsilon = 1 \end{cases}\right\} & \text{if} \quad \varepsilon' = -1 \end{cases} \tag{5.10}$$

Consequently,

$$f_\varepsilon(\varepsilon) = \begin{cases} p_1 \dfrac{1-c}{2} + p_2 \dfrac{1+c}{2}, & \text{if} \quad \varepsilon = 1 \\ p_1 \dfrac{1+c}{2} + p_2 \dfrac{1-c}{2}, & \text{if} \quad \varepsilon = -1 \end{cases}$$

Replacing ε with ε' in the joint distribution (5.3), we obtain the following distribution:

$$f(\mathbf{x}, \varepsilon) = \begin{cases} p_1 \dfrac{1-bc}{2} f_1(\mathbf{x}) + p_2 \dfrac{1+bc}{2} f_2(\mathbf{x}), & \text{if} \quad \varepsilon = 1 \\ p_1 \dfrac{1+bc}{2} f_1(\mathbf{x}) + p_2 \dfrac{1-bc}{2} f_2(\mathbf{x}), & \text{if} \quad \varepsilon = -1 \end{cases} \tag{5.11}$$

The derivation of (5.11) is done in the work [5-2] where the analysis of particular and general cases is performed. It is shown that in the case of different "teacher's slant about his abilities" concerning the first and second pattern classes,

$$f(\mathbf{x},\varepsilon)=\begin{cases} p_1 f_1(\mathbf{x})\dfrac{2+(c_2-c_1)-b_1(c_1+c_2)}{4}+p_2 f_2(\mathbf{x})\dfrac{2+(c_2-c_1)+b_2(c_1+c_2)}{4}, & \text{if } \varepsilon=1 \\ p_1 f_1(\mathbf{x})\dfrac{2+(c_2-c_1)-b_1(c_1+c_2)}{4}+p_2 f_2(\mathbf{x})\dfrac{2+(c_2-c_1)-b_2(c_1+c_2)}{4}, & \text{if } \varepsilon=-1 \end{cases}$$

The expressions for eigendistribution, conditional distributions and distribution moments for the input signal can be obtained from the latter equation.

5.3 Joint Distribution Law for the Input Signal Probabilities in the Case of *K* Classes of Patterns

The matrix of probability $a_{kk'}$ that the teacher will consider a particular pattern as belonging to the k-th and k'-th class is introduced a priori when the number of classes is more than two:

$$A=\begin{bmatrix} a_{11}\cdots a_{1k'}\cdots a_{1K} \\ \vdots\ \ddots\ \vdots\ \ddots\ \vdots \\ a\ \cdots a_{kk'}\cdots a_{kK} \\ \vdots\ \ddots\ \vdots\ \ddots\ \vdots \\ a_{K1}\cdots a_{KK'}\cdots a_{KK} \end{bmatrix}$$

It is evident that

$$\sum_{k=1}^{K} a_{kk'}=1 \quad , \quad k,k'=1,\ldots,K$$

The joint probability distribution law for signals $\mathbf{x}(n)$ and $\boldsymbol{\varepsilon}(n)$ has the form

$$f(\mathbf{x},\varepsilon)=\sum_{k'=1}^{K} p_k a_{kk'} f_{k'}(\mathbf{x}) \quad , \quad \text{if} \quad \varepsilon=k \tag{5.12}$$

where $k=1,\ldots,K$.

In the learning mode, matrix A is a unit matrix

$$A_1=\begin{bmatrix} 1\ 0\ \ldots\ 0 \\ \ldots\ldots\ldots\ldots \\ 0\ 1\ \ldots\ 0 \\ \ldots\ldots\ldots\ldots \\ 0\ 0\ \ldots\ 1 \end{bmatrix}$$

In the self-learning mode, the probability of considering the k'-class pattern as belonging to the k-th class is equal to $1/K$ for all classes:

$$A_2 = \begin{bmatrix} \frac{1}{K} \cdots \frac{1}{K} \\ \cdots\cdots \\ \frac{1}{K} \cdots \frac{1}{K} \end{bmatrix}$$

In the "sabotage" mode, k'-class patterns are considered with some probability of belonging to any class except this k'-class:

$$A_3 = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1K} \\ \cdots & \cdots & \cdots & \cdots \\ a_{21} & 0 & \cdots & a_{2K} \\ \cdots & \cdots & \cdots & \cdots \\ a_{K1} & a_{K2} & \cdots & 0 \end{bmatrix}$$

Let us introduce the notion of the teacher qualification b_k for the pattern recognition system in the case of K classes. The relationship between probabilities $a_{kk'}$ and b_k is nonlinear because

$$b_k = \begin{cases} 1\ , & \text{if} \quad a_{kk} = 1 \\ 0\ , & \text{if} \quad a_{kk} = \frac{1}{K} \\ -1\ , & \text{if} \quad a_{kk} = 0 \end{cases} \tag{5.13}$$

If this relationship were approximated by the function of the second order

$$a_{kk} = x b_k^2 + y b_k + z$$

then after the substitution of (5.13) one obtains

$$a_{kk} = \left[\frac{1}{2} - \frac{1}{K}\right] b_k^2 + \frac{1}{2} b_k + \frac{1}{K} \tag{5.14}$$

Similarly, the relationship $b(a)$ has the form

$$b_{kk} = \frac{K(K-2)}{1-K} a_{kk}^2 + \frac{2-K^2}{1-k} a_{kk} - 1 \tag{5.15}$$

Any of the expressions of (5.14) or (5.15) can be used in concrete calculations. The expression (5.12) allows one to obtain the final expression for the distribution moment:

$$\overline{\varepsilon^i \mathbf{x}^j}^n = \sum_{k=1}^{K} \sum_{k'=1}^{K} p_{k'} a_{kk'} a_{jk'} k^i$$

The joint probability distribution for the input signal in the case of K pattern classes, arbitrary teacher qualification and "teacher's slant about its abilities" with respect to each class has the form

$$f(\mathbf{x},\varepsilon)=\sum_{k=1}^{K} c_{lk}\sum_{k'=1}^{K} a_{kk'}p_{k'}f_{k'}(\mathbf{x}) \quad , \quad \text{if} \quad \varepsilon=l \quad , \quad l=1,\ldots,K \, ,$$

where the probability matrix $C=[c_{lk}]$ characterizes the "teacher's slant about its abilities" when considering l-class patterns as k-class ones. If the following is designated

$$\sum_{k=1}^{K} c_{lk}a_{kk'}=d_{lk'} \sum\ldots=d_{ik}$$

then one obtains

$$f(\mathbf{x},\varepsilon)=\sum_{k'=1}^{K} p_{k'}f_{k'}(\mathbf{x})d_{lk} \quad , \quad \text{if} \quad \varepsilon=l, \;\; l=1,\ldots,K \, .$$

The case of a pattern class continuum. The eigendistribution of supervisor instructions in the learning mode for recognition of K classes has the form

$$f_{\varepsilon}(\varepsilon) = p_k, \text{ when } \varepsilon = k, \; k = 1,\ldots,K$$

This is a function of discrete argument ε. The case of indiscrete distribution function has a wide practical use when the neural network teacher cannot clearly determine the patterns belonging to a particular class. It is possible (but not desirable due to the loss of information) to partition the axis T into K parts and to reduce the problem with a class continuum to the problem with K pattern classes. In the case of continuum pattern classes, unit matrix A and continuous function $f_{\varepsilon}(\varepsilon)$ in the learning mode, one obtains similar to (5.12) the following:

$$f(\mathbf{x},\varepsilon) = f_{\varepsilon}(\varepsilon)f'(\mathbf{x}/\varepsilon)$$

and in the self-learning mode,

$$f(\mathbf{x},\varepsilon) = f_{\varepsilon}(\varepsilon)f_{\mathrm{x}}(\mathbf{x})$$

A function $a(\varepsilon',\varepsilon)$ of the probability of considering patterns objectively corresponding to the distribution $f(\mathbf{x},\varepsilon')$ as corresponding to the distribution $f(\mathbf{x},\varepsilon)$ is introduced in the case of the arbitrary teacher qualification. Then

$$\int_{-\infty}^{\infty} a\left(\varepsilon',\varepsilon\right)\mathrm{d}\varepsilon' = 1$$

The joint probability distribution $f(\mathbf{x},\varepsilon)$ of the neural network input signals $\mathbf{x}(n)$ and $\varepsilon(n)$ will have the form

$$f(\mathbf{x},\varepsilon)= \int_{-\infty}^{\infty} a\left(\varepsilon',\varepsilon\right) f(\mathbf{x},\varepsilon')\mathrm{d}\varepsilon'$$

For the learning mode, $a(\varepsilon',\varepsilon) = \delta(\varepsilon'\text{-}\varepsilon)$ and $f(\mathbf{x},\varepsilon) = f(\mathbf{x},\varepsilon)$.

About nonstationary neural network input signals. Assemblages of patterns distributed inside each class according to the time dependent law $f_k(\mathbf{x},n)$ are considered in the case of nonstationary input signals. The change of distribution $f(\mathbf{x},\varepsilon)$ with time n can take place due to the change of conditional densities $f'(\mathbf{x}/\varepsilon)$ or distribution of supervisor instructions $f_\varepsilon(\varepsilon)$. The general expression for the joint distribution law of the neural network input signal has the form

$$f(\mathbf{x},\varepsilon,n)=\sum_{k'=1}^{K} d_{lk'}p_{k'}f_{k'}(\mathbf{x},n) \quad , \quad \text{if} \quad \varepsilon=l \quad , \quad l=1,\ldots,K$$

In principle, one can also consider the more general case with time-dependent teacher qualification and "teacher's slant about its abilities". The expression for the moments of such a distribution at the current time has the form

$$\overline{\varepsilon^i \mathbf{x}^j}^n = \sum_{k=1}^{K}\sum_{k'=1}^{K} d_{k'k}(n)\alpha_{jk'}(n)k^i$$

We describe in this chapter the analysis of distribution functions for the neural network input signal in the case of arbitrary teacher qualification. In particular, we consider learning, self-learning and "sabotage" as well as some intermediate cases. Generally, the teacher can instruct the neural network in the form of a multidimensional vector $\varepsilon(n)$ with dimensionality N. The formal expressions for the input signal distribution function in the majority of cases are the same. The expressions for the input signal distribution laws can be written in the general form relatively a priori probability of class appearance and conditional distributions $f'(\mathbf{x}/\varepsilon)$.

Literature

[5-1] Galushkin AI (1970) Unified approach to the problem of learning and self-learning for the pattern recognition systems. MIEM proc. 6:104–120

[5–2] Galushkin AI (1971) About input signal characteristics for the pattern recognition systems. MIEM proc, 14:125–138

Chapter 6

Design of Neural Network Optimal Models

6.1 General Structure of the Optimal Model

The optimal neural network model is such optimal transformation of the input signal $[\mathbf{x}(n), \boldsymbol{\varepsilon}(n)]$ and acquisition of the output signal $\mathbf{y}(n)$ according to the selected primary optimization criterion (Fig. 6.1).

The upper unit in Fig. 6.1 represents a controlled system, and the lower unit represents an optimal model that adjusts the controlled system. The general approach for the design of the optimal models of a pattern recognition system in the learning mode consists in the learning arbitrary characteristics of the input signal that represents two, K and continuum classes of patterns. The number of the neural network solutions is also arbitrary. The solution space has two, K and a continuum of gradations. The supervisor instructions and solution characteristics are selected a priori and independently.

The design of the optimal model is performed according to the selected criterion of primary optimization. The model description is performed in the form of a divisional surface. The divisional surface partitions the multidimensional feature space into nonoverlapping areas with the instruction of the corresponding area belonging to some class.

Table 6.1 represents classification of neural networks according to the input signal characteristics and solution space for a particular form of one-dimensional signals $y(n)$ and $\varepsilon(n)$.

System of the first type with two pattern classes with binary output is most widely used.

System of the second type with K pattern classes and with number of solutions equal to K. The investigation of such a system is given in [6-1 to 6-3] for different criteria of primary optimization and different a priori information about input signal characteristics. The idea that the problem of learning during recognition of K pattern classes can be reduced to the sequential step-by-step use of a learning algorithm for two classes

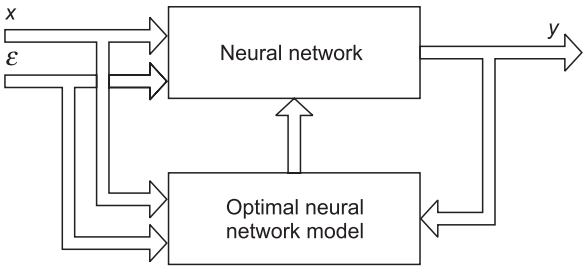


Fig. 6.1.
To the optimal model definition

Table 6.1. Classification of neural networks

Solution space (number)	Input signal				
	Two classes		*K* classes		Continuum of classes
Two	1		7		8
K_p	$K_p = 3$	3a	$K < K_p$	9	10
			$K = K_p$	2	
	K_p = const.	3b	$K > K_p$	4	
Continuum	5		6		11

is not correct. This is a separate problem whose optimal solution requires the generation of an equivalent divisional surface starting from the first step.

The neural network solution space is characterized by the number of amplitude quantization levels for the output signal $\mathbf{y}(n)$ in each channel. The neural networks with a continuum of solutions (5 or 6, Table 6.1) have an indiscrete output signal. The neural network of 8, 10 or 11 type has an indiscrete distribution of supervisor instructions.

6.2 Analytical Representation of Divisional Surfaces in Typical Neural Networks

Methods of optimal neural network model designs for *K* pattern class recognition are given in [6-1 to 6-3]. The expressions for optimal divisional surfaces can be obtained from the expression of a minimized functional of primary optimization and a solution of the minimization problem for this functional with the existent limitations.

The optimal neural network model is determined by the system of inequalities for the initial space partitioning into *K* areas. Let us consider the design of the optimal neural network models shown in Table 6.1.

Neural network of the third type. The system of pattern recognition optimal by the maximum posterior probability (in the case of two solutions) transforms the input signal $\mathbf{x}(n)$ into the output signal $\mathbf{y}(n)$ according to the following expression

$$y(n) = \begin{cases} 1, & \text{if} \quad f(\varepsilon = 1/\mathbf{x}) > f(\varepsilon = -1/\mathbf{x}) \\ -1, & \text{if} \quad f(\varepsilon = -1/\mathbf{x}) > f(\varepsilon = 1/\mathbf{x}) \end{cases}$$

The divisional surface is put through those points $\mathbf{x}$ that have equal posterior probabilities of belonging to the first and second classes. The multidimensional space area with higher posterior probability of belonging to the first class is taken as the area of the first class. However, points belonging to some class in a multidimensional feature space must be indicated with more definite probability in the majority of practical tasks. For the system of the first type (one divisional surface), this probability decreases as the points approach the divisional surface and becomes zero at the surface.

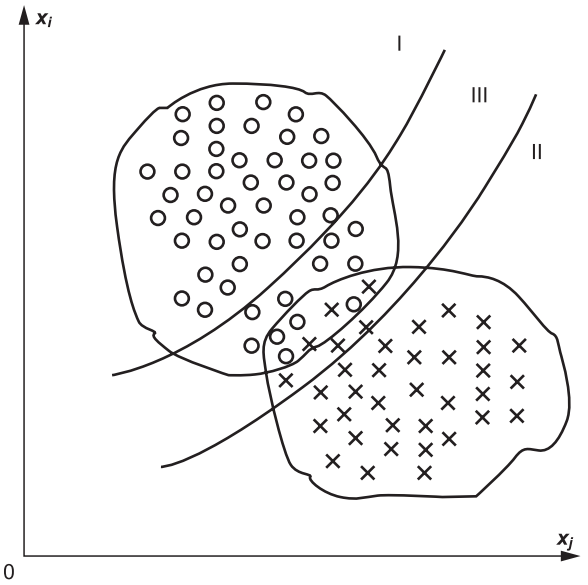


Fig. 6.2.
The partition of the feature space by two divisional surfaces

The system of pattern recognition of the 3*a*-type has two divisional surfaces. They partition the feature space into three parts (I, II, III, Fig. 6.2) with inactive regions in which the system determines the input pattern: region I – to the first class, region II – belonging to the second class, and region III – when the current pattern cannot be classified.

The multidimensional feature space must be divided into three regions:

- Region I corresponds to the neural network decision about its belonging to the first class

$$f''(\varepsilon = -1/\mathbf{x}) - d_1 > f''(\varepsilon = 1/\mathbf{x})$$

- Region II corresponds to the neural network decision about its belonging to the second class

$$f''(\varepsilon = -1/\mathbf{x}) + d_2 < f''(\varepsilon = 1/\mathbf{x})$$

- Region III corresponds to the case when the neural network cannot make a decision about the current pattern belonging to the first or second classes

$$f''(\varepsilon = -1/\mathbf{x}) - d_1 < f''(\varepsilon = 1/\mathbf{x})$$

$$f''(\varepsilon = -1/\mathbf{x}) + d_2 > f''(\varepsilon = 1/\mathbf{x})$$

Here parameters d_1 and d_2 ($0 \le d_1 \le 1, 0 \le d_2 \le 1$) determine the probability level of pattern consideration belonging to the first or second classes. It is possible, in

particular, that $d_1 = d_2 = d$ or $d_1 = d_2 = d = 0$. In the latter case, two divisional surfaces are reduced to one surface. For two divisional surfaces, the pattern recognition system with optimal surface parameters transforms the input signal $\mathbf{x}(n)$ into the output signal $y(n)$ in the following way: in the region I $y(n) = -1$ (first class); in the region II $y(n) = 1$ (second class); in the region III $y(n) = 0$ (first and second classes).

The general expression for the divisional surfaces optimal by the posterior probability value has the form

$$\frac{p_1 f_1(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})} - d_1 = \frac{p_2 f_2(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})}$$

$$\frac{p_1 f_1(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})} + d_2 = \frac{p_2 f_2(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})}.$$

One obtains after transformations

$$\left.\begin{aligned} S'(\mathbf{x}) &= \frac{f_2(\mathbf{x})}{f_1(\mathbf{x})} - \frac{(1-d_1)p_1}{(1+d_1)p_2} \\ S''(\mathbf{x}) &= \frac{f_2(\mathbf{x})}{f_1(\mathbf{x})} - \frac{(1+d_2)p_1}{(1+d_2)p_2} \end{aligned}\right\} \tag{6.1}$$

This is the final expression for the divisional surface optimum by the posterior probability value as the primary optimization criterion. A more detailed interpretation is given in [6-1, 6-2].

The pattern recognition system optimum by minimum average risk function criterion partitions the multidimensional feature space into three regions: first and second classes of patterns belonging, and the region where the neural network cannot make a decision:

$$\left.\begin{aligned} &S'(\mathbf{x}) < 0 \\ &S''(\mathbf{x}) > 0 \\ &S''(\mathbf{x}) < 0 < S'(\mathbf{x}) \end{aligned}\right\} \tag{6.2}$$

Conditional risk function is a sum of losses due to consideration of the i-th class patterns as patterns of j-th class. The losses are calculated as corresponding probabilities multiplied by coefficients l_{ij} ($i = 1, 2$; $j = 1, 0, 2$) of the loss matrix L

$$L = \begin{bmatrix} l_{11} & l_{10} & l_{12} \\ l_{21} & l_{20} & l_{22} \end{bmatrix}$$

Coefficients l_{10} and l_{20} are the loss coefficients when the recognition system cannot make a decision. It is evident that

$$l_{11} < l_{10} < l_{12};\ l_{21} > l_{20} > l_{22}$$

The expression for the conditional risk function has the following form:

$$r_1 = \overbrace{\int_{S'(\mathbf{x})<0} \dots \int}^{N} l_{11} f_1(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{S''(\mathbf{x})<0<S'(\mathbf{x})} \dots \int}^{N} l_{10} f_1(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{S''(\mathbf{x})>0} \dots \int}^{N} l_{12} f_1(\mathbf{x})\mathrm{d}\mathbf{x} \tag{6.3}$$

$$r_2 = \overbrace{\int_{S'(\mathbf{x})<0} \dots \int}^{N} l_{21} f_2(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{S''(\mathbf{x})<0<S'(\mathbf{x})} \dots \int}^{N} l_{20} f_2(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{S''(\mathbf{x})>0} \dots \int}^{N} l_{22} f_2(\mathbf{x})\mathrm{d}\mathbf{x} \tag{6.4}$$

After averaging of conditional risk functions, one obtains average risk functions

$$R = \overbrace{\int_{S'(\mathbf{x})<0} \dots \int}^{N} [l_{11} p_1 f_1(\mathbf{x}) + l_{21} p_2 f_2(\mathbf{x})]\mathrm{d}\mathbf{x} + \overbrace{\int_{S''(\mathbf{x})<0<S'(\mathbf{x})} \dots \int}^{N} [l_{10} p_1 f_1(\mathbf{x}) + l_{20} p_2 f_2(\mathbf{x})]\mathrm{d}\mathbf{x}$$
$$+ \overbrace{\int_{S''(\mathbf{x})>0} \dots \int}^{N} [l_{12} p_1 f_1(\mathbf{x}) + l_{22} p_2 f_2(\mathbf{x})]\mathrm{d}\mathbf{x}$$

Taking into account that

$$\overbrace{\int_{S''(\mathbf{x})<0<S'(\mathbf{x})} \dots \int}^{N} = \overbrace{\int_{S''(\mathbf{x})<0} \dots \int}^{N} - \overbrace{\int_{S'(\mathbf{x})<0} \dots \int}^{N}$$

$$\overbrace{\int_{S''(\mathbf{x})>0} \dots \int}^{N} = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} = \overbrace{\int_{S''(\mathbf{x})<0} \dots \int}^{N}$$

and

$$\overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} [l_{12} p_1 f_1(\mathbf{x}) + l_{22} p_2 f_2(\mathbf{x})]\mathrm{d}\mathbf{x} = l_{12} p_1 + l_{22} p_2$$

the expression for the average risk function takes the form

$$R = (l_{12} p_1 + l_{22} p_2) + \overbrace{\int_{S'(\mathbf{x})<0} \dots \int}^{N} [l_{11} p_1 f_1(\mathbf{x}) + l_{21} p_2 f_2(\mathbf{x}) - l_{10} p_1 f_1(\mathbf{x}) - l_{20} p_2 f_2(\mathbf{x})]\mathrm{d}\mathbf{x}$$
$$+ \overbrace{\int_{S''(\mathbf{x})<0} \dots \int}^{N} [l_{10} p_1 f_1(\mathbf{x}) + l_{20} p_2 f_2(\mathbf{x}) - l_{12} p_1 f_1(\mathbf{x}) - l_{22} p_2 f_2(\mathbf{x})]\mathrm{d}\mathbf{x}$$

Consequently, the final expression for the average risk function is

$$R=(l_{12}p_1+l_{22}p_2)+\overbrace{\int\limits_{S'(\mathbf{x})<0}\cdots\int}^{N}[(l_{11}-l_{10})p_1f_1(\mathbf{x})+(l_{21}-l_{20})p_2f_2(\mathbf{x})f_2(\mathbf{x})]\mathrm{d}\mathbf{x}$$
$$+\overbrace{\int\limits_{S''(\mathbf{x})<0}\cdots\int}^{N}[(l_{10}-l_{12})p_1f_1(\mathbf{x})+(l_{20}-l_{22})p_2f_2(\mathbf{x})]\mathrm{d}\mathbf{x} \tag{6.5}$$

It is necessary to derivate the expressions for $S'(\mathbf{x})$ and $S''(\mathbf{x})$ that provide the minimum of R. It is relatively simple to show that this minimum is achieved when the expressions under the integral sign are negative in the corresponding region and are positive outside it. Consequently, the minimum of R is achieved when

$$\left.\begin{aligned}S'(\mathbf{x})&=(l_{11}-l_{10})\,p_1f_1(\mathbf{x})+(l_{21}-l_{20})\,p_2f_2(\mathbf{x})\\S''(\mathbf{x})&=(l_{10}-l_{12})\,p_1f_1(\mathbf{x})+(l_{20}-l_{22})\,p_2f_2(\mathbf{x})\end{aligned}\right\} \tag{6.6}$$

The expressions (6.2) and (6.6) determine the optimal neural network model for the recognition of two pattern classes with two divisional surfaces. Let us assume

$$l_{11}=l_{22}=0,\ l_{12}=l_{21}=1,\ l_{10}=l_{20}=l_0,\ p_1=p_2$$

and

$$S'(\mathbf{x})=(1-l_0)\,f_2(\mathbf{x})-l_0f_1(\mathbf{x})$$

$$S''(\mathbf{x})=l_0f_2(\mathbf{x})-(1-l_0)\,f_1(\mathbf{x})$$

Figure 6.3 illustrates the changes of thresholds h_1 and h_2 depending on l_0.

The analysis of expressions for divisional surfaces provides the following conclusions:

a If $l_0=0$ then the region of the neural network where it cannot make a decision occupies all of the feature space because the losses are zero;
b If $l_0=0.5$ then the neural network with two divisional surfaces is reduced to the neural network with one divisional surface. In this case, the losses without recognition are two times fewer than with wrong recognition. The losses with correct recognition are zero;
c If $0.5>l_0>0$ then an inactive region exists where the neural network does not consider the current patterns as belonging to any class;
d If $1>l_0>0.5$ then the neural network realizes two divisional surfaces. The curves in Fig. 6.3 for the change of thresholds are symmetrical both relative to the line $f_1(\mathbf{x})=f_2(\mathbf{x})$ as well as relative to the level $l_0=0.5$. The threshold h_1 determines the surface $S'(\mathbf{x})$ and the threshold h_2 determines the surface $S''(\mathbf{x})$;
e If $l_0=1$ then all the multidimensional feature space is considered to belong both to the first and second classes.

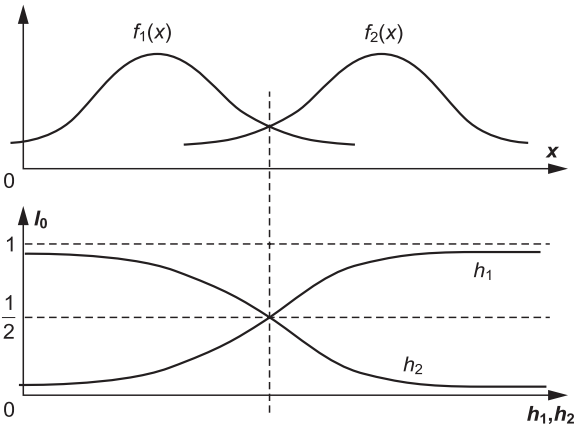


Fig. 6.3.
The investigation of the neural network structure with dependence on the loss coefficient under the system recognition refusal

The comparison between optimal models designed by a posterior probability criterion (6.1) and minimum risk function criterion (6.5) shows that in the case when

$$d_1 = \frac{(l_{11} + l_{21}) - (l_{10} + l_{20})}{2} \quad ; \quad d_2 = \frac{(l_{12} + l_{22}) - (l_{10} + l_{20})}{2}$$

the optimal models coincide. This gives the additional interpretation of coefficients d_1 and d_2. Both criteria can be used when the a priori information about d_j or l_{ij} is known.

The analysis of expression for the average risk function shows the possibility of considering the primary optimization criteria under the following limitations:

1. The equality of the separate average components of the risk function

$$p_1 r_1 = p_2 r_2 \tag{6.7}$$

2. Constant component of the average risk function

$$p_2 r_2 = \alpha = \text{const.} \tag{6.8}$$

Let us write the optimization Lagrange functional for the first limitation in the form

$$I = R + \lambda (p_1 r_1 - p_2 r_2)$$

One obtains from (6.7), (6.3) and (6.4)

$$\begin{aligned} &p_1 l_{12} + \overbrace{\int\limits_{S'(\mathbf{x})<0} \cdots \int}^{N} (l_{11} - l_{10}) p_1 f_1(\mathbf{x}) d\mathbf{x} + \overbrace{\int\limits_{S''(\mathbf{x})<0} \cdots \int}^{N} (l_{10} - l_{12}) p_1 f_1(\mathbf{x}) d\mathbf{x} \\ &= p_2 l_{22} + \overbrace{\int\limits_{S'(\mathbf{x})<0} \cdots \int}^{N} (l_{21} - l_{20}) p_2 f_2(\mathbf{x}) d\mathbf{x} + \overbrace{\int\limits_{S''(\mathbf{x})<0} \cdots \int}^{N} (l_{20} - l_{22}) p_2 f_2(\mathbf{x}) d\mathbf{x} \end{aligned} \tag{6.9}$$

The equations for the optimal divisional surfaces

$$\left.\begin{aligned} S'(\mathbf{x}) &= (l_{11}-l_{10})\,p_1 f_1(\mathbf{x})(1+\lambda)+(l_{21}-l_{20})\,p_2 f_2(\mathbf{x})(1-\lambda) \\ S''(\mathbf{x}) &= (l_{10}-l_{12})\,p_1 f_1(\mathbf{x})(1+\lambda)+(l_{20}-l_{22})\,p_2 f_2(\mathbf{x})(1-\lambda) \end{aligned}\right\} \tag{6.10}$$

are the result of functional I minimization. The value λ is obtained from (6.9) and (6.10) for corresponding limitation.

In the case of limitation (6.8), the average risk function minimization criterion gives the following form of the optimal divisional surfaces

$$\left.\begin{aligned} S'(\mathbf{x}) &= (l_{11}-l_{10})\,p_1 f_1(\mathbf{x})+\lambda(l_{21}-l_{20})\,p_2 f_2(\mathbf{x}) \\ S''(\mathbf{x}) &= (l_{10}-l_{12})\,p_1 f_1(\mathbf{x})+\lambda(l_{20}-l_{22})\,p_2 f_2(\mathbf{x}) \end{aligned}\right\} \tag{6.11}$$

The expression for λ is obtained from (6.11) and corresponding limitation

$$p_2 r_2 = p_2 l_{22} + \overbrace{\int\limits_{S'(\mathbf{x})<0}\!\!\cdots\!\!\int}^{N} (l_{21}-l_{20}) p_2 f_2(\mathbf{x})\,\mathrm{d}\mathbf{x} + \overbrace{\int\limits_{S'(\mathbf{x})<0}\!\!\cdots\!\!\int}^{N} (l_{20}-l_{22}) p_2 f_2(\mathbf{x})\,\mathrm{d}\mathbf{x} = \alpha$$

Neural network of the 3b type. Let us consider the neural network of the *3b* type (Table 6.1) for two patterns with $(K_p - 1)$ divisional surfaces. K_p = const. means that the integer number of the solution amounts to 4 or higher.

Let us consider the neural network optimal model by the posterior probability criterion. The multidimensional space partition is determined in the following way:

The region k_p ($k_p = 1, \ldots, K_p$) is determined by the following system of inequalities:

$$f(\varepsilon=-1/\mathbf{x}) - d_{k_p-1,k_p} < f(\varepsilon=1/\mathbf{x}) < f(\varepsilon=-1/\mathbf{x}) - d_{k_p,k_p+1}$$

under conditions $d_{0,1} = 1$, $d_{Kp,\ Kp+1} = -1$, and

$$d_{k_p,k_p+1} > 0 \quad , \quad \text{if} \quad f(\varepsilon=-1/\mathbf{x}) > f(\varepsilon=1/\mathbf{x})$$

$$d_{k_p,k_p+1} < 0 \quad , \quad \text{if} \quad f(\varepsilon=-1/\mathbf{x}) < f(\varepsilon=1/\mathbf{x})$$

Figure 6.4 shows the example of such partitioning in a one-dimensional case. The neural network output signal must have K_p level gradations, i.e., at two pattern classes, the neural network makes K_p decisions. The decisions are made with some margins by the posterior probability, as it is seen from Fig. 6.4.

Using the known expressions for posterior probabilities $f(\varepsilon = -1/\mathbf{x})$ and $f(\varepsilon = 1/\mathbf{x})$, one can obtain the expression for the k_p-th solution region in the initial feature space:

$$\frac{1-d_{k_p-1,k_p}}{1+d_{k_p-1,k_p}} < \frac{p_2 f_2(\mathbf{x})}{p_1 f_1(\mathbf{x})} < \frac{1-d_{k_p,k_p+1}}{1+d_{k_p,k_p+1}}$$

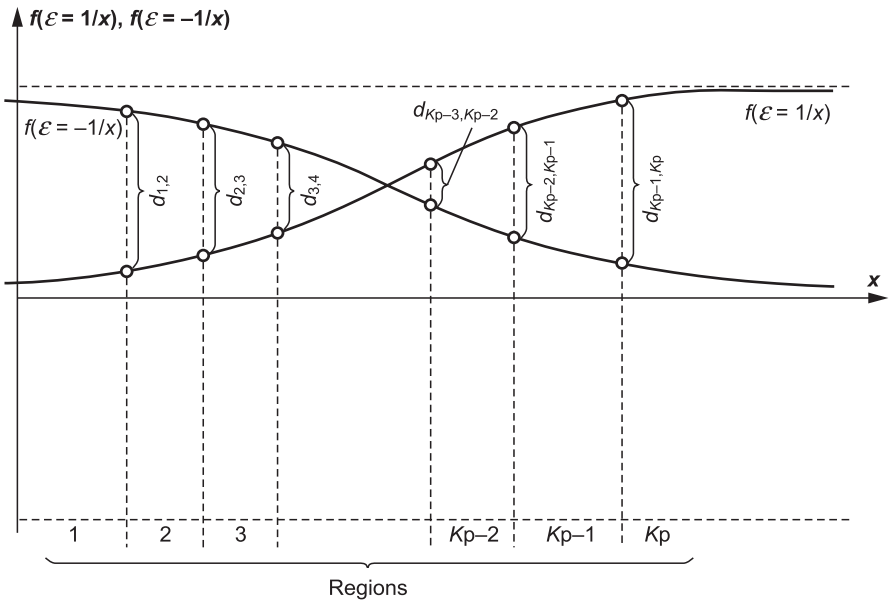


Fig. 6.4. The consideration of the posterior probability primary optimization criterion in the case of two pattern classes and $(K_p - 1)$ divisional surfaces

Let us determine the optimal neural network model by the minimum average risk function criterion. After the learning procedure, the neural network partitions the multidimensional feature space into K_p regions with a priori losses in each of them. The matrix of loss coefficients has the form

$$L = \begin{bmatrix} l_{11} & l_{12} & \dots & l_{1K_p} \\ l_{21} & l_{22} & \dots & l_{2K_p} \end{bmatrix} \tag{6.12}$$

where l_{ik_p} $(i = 1, 2;\ k_p = 1, \dots, K_p)$ are the loss coefficients for the consideration of the i-th pattern class as belonging to the k_p-th region. It is necessary that

$$l_{11} < l_{12} < \dots < l_{1K_p} \quad ; \quad l_{21} > l_{22} > \dots > l_{2K_p}$$

The expressions for conditional risk functions have the following forms

$$r_1 = \sum_{k_p=1}^{K_p} \overbrace{\int \dots \int}^{N}_{S^{k_p}(\mathbf{x})>0} l_{1k_p} f_1(\mathbf{x}) \mathrm{d}\mathbf{x}$$

$$r_2 = \sum_{k_p=1}^{K_p} \overbrace{\int \dots \int}^{N}_{S^{k_p}(\mathbf{x})>0} l_{2k_p} f_2(\mathbf{x}) \mathrm{d}\mathbf{x}$$

Here $S^{k_p}(\mathbf{x}) > 0$ is the region of the multidimensional feature space corresponding to the k_p-th decision. Consequently, the expression for the average risk function is

$$R = \sum_{k_p=1}^{K_p} \overbrace{\int \dots \int}^{N}_{S^{k_p}(\mathbf{x})>0} \left[l_{1k_p} p_1 f_1(\mathbf{x}) + l_{2k_p} p_2 f_2(\mathbf{x}) \right] d\mathbf{x}$$

Let us find the expression for $S^{k_p}(\mathbf{x})$ that minimizes the average risk function. With the following additional notion

$$g_{k_p}(\mathbf{x}) = \left[l_{1k_p} p_1 f_1(\mathbf{x}) + l_{2k_p} p_2 f_2(\mathbf{x}) \right]$$

one obtains

$$R = \sum_{k_p=1}^{K_p} \overbrace{\int \dots \int}^{N}_{S^{k_p}(\mathbf{x})>0} g_{k_p}(\mathbf{x}) d\mathbf{x}$$

It can be shown using the given expressions above that minimum R is achieved at

$$S^{(k_p)}(\mathbf{x}) = g_{k_p''}(\mathbf{x}) - g_{k_p}(\mathbf{x}) > 0 \quad , \quad k_p'' = 1, \dots, K_p$$

or

$$S^{(k_p)}(\mathbf{x}) = l_{1k_p''} p_1 f_1(\mathbf{x}) + l_{2k_p''} p_2 f_2(\mathbf{x}) - l_{1k_p} p_1 f_1(\mathbf{x}) - l_{2k_p} p_2 f_2(\mathbf{x}) < 0 \quad , \quad k_p'' = 1, \dots, K_p$$

The expressions for other aforementioned optimization criteria can be derived similarly.

Neural networks of types 4 and 9. This is the case of the neural network with K pattern class recognition and with $(K_p - 1)$ divisional surfaces.

The system of inequalities in the case of minimum average risk function in the region of k_p-th decision has the following form:

$$S^{(k_p)}(\mathbf{x}) = \sum_{k=1}^{K} \left(l_{kk_p} - l_{kk_p''} \right) p_k f_k(\mathbf{x}) < 0 \quad , \quad k_p'' = 1, \dots, K_p$$

Neural network of type 5. The neural network has an indiscrete amplitude output signal with two input pattern classes. However, the input and output signals are discrete in time.

One must have an a priori function $d(y)$ for the probability to exceed in the case of the use of a posterior probability primary optimization criterion. The equation for the neural network optimal model with two pattern classes and a continuum of solutions has the form

$$f''(\varepsilon = -1/\mathbf{x}) - d(y) = f''(\varepsilon = 1/\mathbf{x})$$

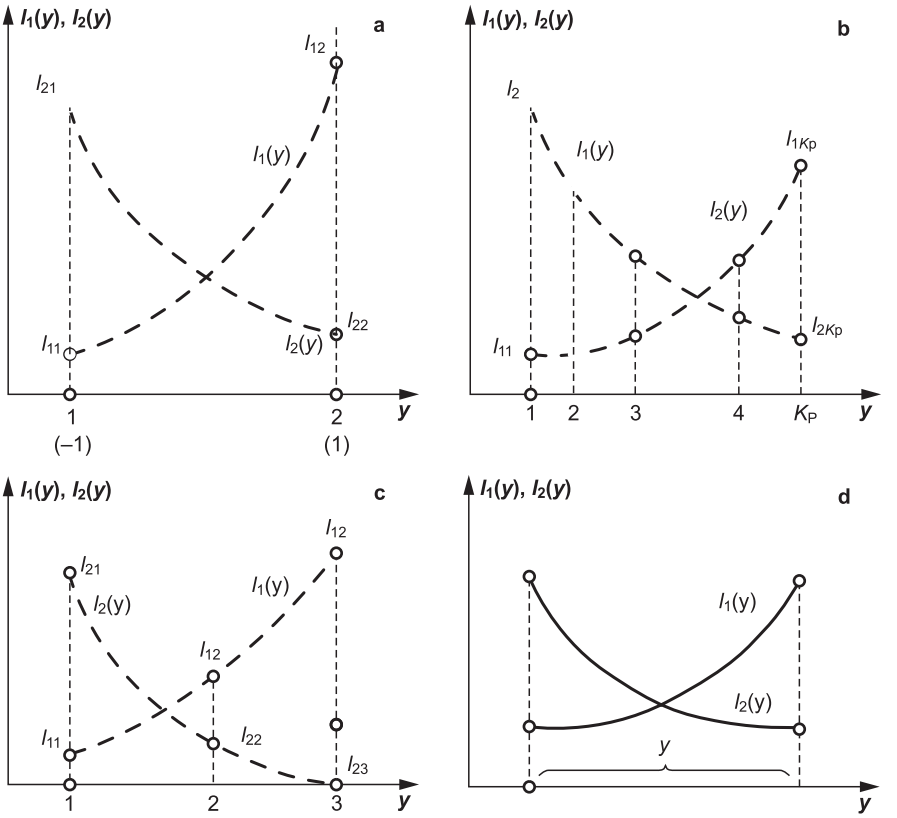


Fig. 6.5. The change of error function during transfer from two neural network solutions to K_p or continuum number of solutions with two pattern recognition classes: **a** two solutions; **b** three solutions, **c** K_p solutions; **d** continuum of solutions

Consequently,

$$\left.\begin{aligned} &\frac{p_1 f_1(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})} - d(y) = \frac{p_2 f_2(\mathbf{x})}{p_1 f_1(\mathbf{x}) + p_2 f_2(\mathbf{x})} \\ &p_1 f_1(\mathbf{x})\left[1 - d(y)\right] - \left[1 + d(y)\right] p_2 f_2(\mathbf{x}) = 0 \end{aligned}\right\} \tag{6.13}$$

This is a final expression for the neural network optimal model in this case. It determines the relationship between the input and output neural network signals.

Let us consider the minimum average risk function criterion. Figure 6.5 illustrates this variant. One must introduce a vector-function of errors instead of a matrix of coefficients (6.12):

$$L = \begin{bmatrix} l_1(y) \\ l_2(y) \end{bmatrix}$$

These errors take place during making decisions y to consider different patterns as belonging to the first and second classes.

In the case of K_p neural network solutions, the expression for the conditional risk function for the first pattern class has the form

$$r_1 = \sum_{k_p=1}^{K_p} l_{1k_p} \overbrace{\int \dots \int}^{N} f_1(\mathbf{x})\mathrm{d}\mathbf{x}$$
$$S^{k_p}(\mathbf{x})>0$$

Here

$$\bigcup_{k_p=1}^{K_p} S^{k_p}(\mathbf{x}) = X$$

and X is the full multidimensional feature space. Let us introduce additional notions:

$$G\left(\mathbf{x},k_p\right)=\begin{cases}1\ , & \text{if} \quad x\in S^{k_p}(\mathbf{x})>0\\ 0\ , & \text{if} \quad x\notin S^{k_p}(\mathbf{x})>0\end{cases}$$

Then the expression for the conditional risk function r_1 takes the following form:

$$r_1 = \sum_{k_p=1}^{K_p} l_{1k_p} \overbrace{\int \dots \int}^{N} G(\mathbf{x},k_p)(\mathbf{x})\mathrm{d}\mathbf{x}$$
$$X$$

Similar to the previous cases, function $G(\mathbf{x}, k_p)$ is an object of synthesis. It determines the neural network optimal model, i.e., an optimal relationship between neural network input and output signals.

The transfer to the continuum of solutions results in the following expressions for the risk function:

$$r_1 = \int_Y l_1(y) \overbrace{\int \dots \int}^{N} G(\mathbf{x},y) f_1(\mathbf{x})\mathrm{d}\mathbf{x}\,\mathrm{d}y$$
$$X$$

$$R = p_1 r_1 + p_2 r_2 = \int_Y \overbrace{\int \dots \int}^{N} G(\mathbf{x},y)\left[p_1 l_1(y) f_1(\mathbf{x}) + p_2 l_2(y) f_2(\mathbf{x})\right]\mathrm{d}\mathbf{x}\,\mathrm{d}y$$
$$X$$

Introducing the notion

$$g(\mathbf{x},y)=\left[p_1 l_1(y) f_1(\mathbf{x}) + p_2 l_2(y) f_2(\mathbf{x})\right]$$

one obtains the final expression for the average risk function

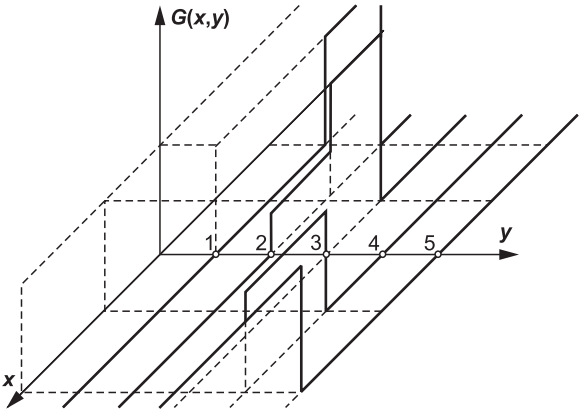


Fig. 6.6.
Form of function $G(\mathbf{x},y)$ for the discrete set of solutions and two pattern classes

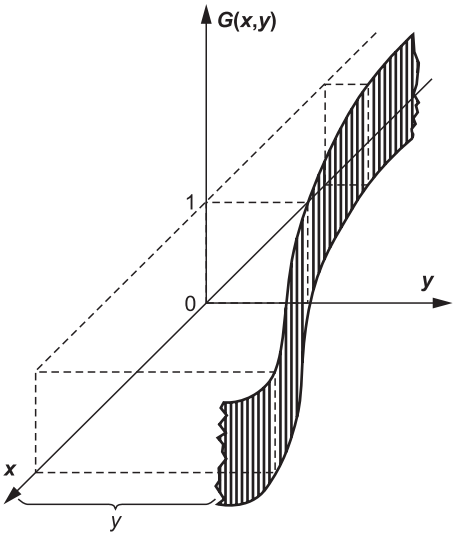


Fig. 6.7.
Form of function $G(\mathbf{x},y)$ for continuum of solutions and two pattern classes

$$R = \int_{Y} \overbrace{\int_{X} \dots \int}^{N} G(\mathbf{x}, y) g(\mathbf{x}, y) \, \mathrm{d}\mathbf{x} \, \mathrm{d}y \tag{6.14}$$

Here function $g(\mathbf{x},y)$ is represented in the general form. The synthesized function $G(\mathbf{x},y)$ must be expressed through the function $g(\mathbf{x},y)$ in such a way that R is minimized. Figure 6.6 shows this function for the one-dimensional case of the feature space and for a finite number K_p of the neural network solutions. In the continuum case, this function is reduced to $G(\mathbf{x},y)$, Fig. 6.7. Its form is a strap with the unit height. The strap shape is an object of synthesis.

Figure 6.8 shows a geometrical illustration of function $G(\mathbf{x},y)$ in the simplest case. Consequently, the problem of the average risk function optimization is reduced to the minimization of the area of the strap $G(\mathbf{x},y)g(\mathbf{x},y)$.

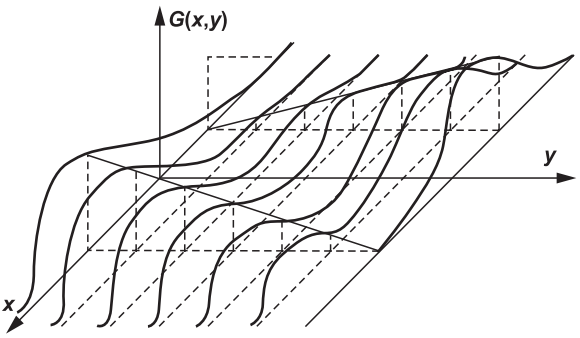


Fig. 6.8.
Form of function $G(\mathbf{x},y)$ for continuum of solutions and two pattern classes

In the expression for the average risk function (6.14),

$$G\left(\mathbf{x},k_{\mathrm{p}}\right)=\begin{cases}1\ , & \text{if} \quad y=p(\mathbf{x})\\ 0\ , & \text{if} \quad y\neq p(\mathbf{x})\end{cases}$$

where function $p(x)$ is an object of synthesis. It is a transformation of the neural network input signal. Consequently, one obtains for $R(\mathbf{x},y)$

$$R=\overbrace{\int_{X}\cdots\int}^{N} g\left[\mathbf{x},P^{*}(x)\right]\mathrm{d}\mathbf{x}$$

where function $g[\mathbf{x},p(\mathbf{x})]$ has the form

$$g\left[\mathbf{x},p(\mathbf{x})\right]=p_1f_1(\mathbf{x})l_1\left[P^{*}(\mathbf{x})\right]+p_2f_2(\mathbf{x})l_2\left[P^{*}(\mathbf{x})\right]$$

Minimization of R is the problem of variational calculation. Its minimum is achieved at

$$\frac{\partial g\left[\mathbf{x},p(\mathbf{x})\right]}{\partial P^{*}(\mathbf{x})}=0$$

For the particular form of $g[\mathbf{x},\,p(\mathbf{x})]$:

$$p_1f_1(\mathbf{x})\frac{\mathrm{d}l_1\left[P^{*}(\mathbf{x})\right]}{\mathrm{d}P^{*}(\mathbf{x})}+p_2f_1(\mathbf{x})\frac{\mathrm{d}l_2\left[P^{*}(\mathbf{x})\right]}{\mathrm{d}P^{*}(\mathbf{x})}=0$$

Or it can be written

$$p_1f_1(\mathbf{x})\left.\frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\right|_{y=P^{*}(\mathbf{x})}+p_2f_2(\mathbf{x})\left.\frac{\mathrm{d}l_2(y)}{\mathrm{d}y}\right|_{y=P^{*}(\mathbf{x})}=0 \tag{6.15}$$

This equation determines the neural network optimal model for two pattern classes and the continuum of solutions.

Let us consider some particular cases.

1. The error function has the form shown in Fig. 6.9. Thus

$$\left.\frac{dl_1(y)}{dy}\right|_{y=P^*(\mathbf{x})} = \left.\sum_{\alpha=1}^{A}\Delta l_{\alpha 1}\delta(y-y_a)\right|_{P^*(\mathbf{x})} = \sum_{\alpha=1}^{A}\Delta l_{\alpha 1}\delta\big(p(\mathbf{x})-y_a\big)$$

$$\left.\frac{dl_2(y)}{dy}\right|_{y=P^*(\mathbf{x})} = \left.\sum_{\alpha=1}^{A}\Delta l_{\alpha 2}\delta(y-y_a)\right|_{P^*(\mathbf{x})} = \sum_{\alpha=1}^{A}\Delta l_{\alpha 2}\delta\big(p(\mathbf{x})-y_a\big)$$

The equation for the neural network optimal model is

$$p_1 f_1(\mathbf{x})\sum_{\alpha=1}^{A}\Delta l_{\alpha 1}\delta(P^*(\mathbf{x})-y_a) - p_2 f_2(\mathbf{x})\sum_{\alpha=1}^{A}\Delta l_{\alpha 2}\delta(P^*(\mathbf{x})-y_a) = 0$$

Here $\delta(y)$ is δ-function with known properties.

2. Error functions for patterns of the first and second classes. These functions are the functions of the second order $l_1(y) = (1+y)^2 l$, $l_2(y) = (1-y)^2 l$.

Hence

$$\frac{dl_1(y)}{dy} = 2l(1+y), \quad \frac{dl_2(y)}{dy} = 2l(y-1)$$

Inserting these expressions into (6.15), one obtains the equation for the neural network optimal model $P^*(\mathbf{x})$ with a continuum of solutions and quadratic losses:

$$P^*(\mathbf{x}) = \frac{p_2 f_2(\mathbf{x}) - p_1 f_1(\mathbf{x})}{p_2 f_2(\mathbf{x}) + p_1 f_1(\mathbf{x})}$$

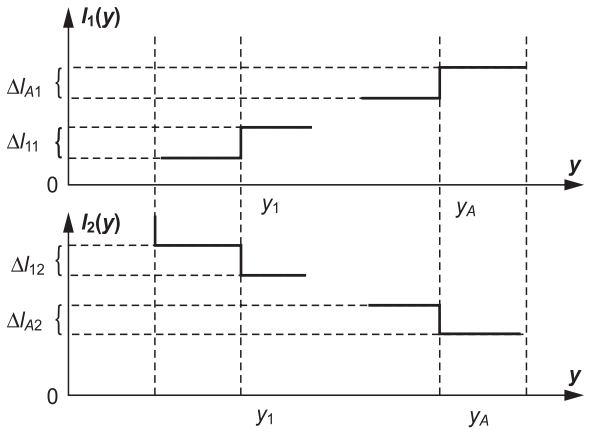


Fig. 6.9.
Error function dependences for the neural network with a continuum of solutions

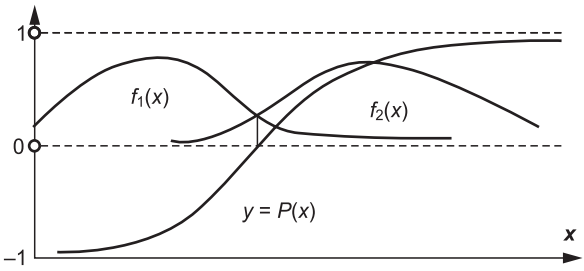


Fig. 6.10.
Illustration of the neural network optimal model with a solution continuum and quadratic losses

Figure 6.10 is the illustration of the function $y = P^*(\mathbf{x})$ realized by the neural network.

3. Error functions of the first order $l_1(y) = (1 + y)l$, $l_2(y) = (1 - y)l$. In this case, $dl_1(y)/dy = l$, $dl_2(y)/dy = -l$. Consequently, some difficulties in the neural network optimal model design arise. Let the error function have the form

$$l_1(y) = l(1+y)^{c+1}, \quad l_2(y) = l(1-y)^{c+1}$$

where $c = 1$ and $c = 0$ correspond to p.2 and p.3, respectively. Then

$$\frac{\mathrm{d}}{\mathrm{d}y} l_1(y) = \frac{\mathrm{d}}{\mathrm{d}y}\left[l(1+y)^{c+1}\right] = \frac{1}{c+1} l(1+y)^c$$

$$\frac{\mathrm{d}}{\mathrm{d}y} l_2(y) = \frac{\mathrm{d}}{\mathrm{d}y}\left[l(1-y)^{c+1}\right] = \frac{1}{c+1} l(1-y)^c$$

A general expression for the neural network optimal model is

$$p_1 f_1(\mathbf{x})(1+y)^c - p_2 f_2(\mathbf{x})(1-y)^c = 0$$

Or in another form,

$$y = \frac{[p_2 f_2(\mathbf{x})]^{1/c} - [p_1 f_1(\mathbf{x})]^{1/c}}{[p_2 f_2(\mathbf{x})]^{1/c} + [p_1 f_1(\mathbf{x})]^{1/c}}$$

The case of p.2 is obtained at $c = 1$. When $c \to 0$ then $y = -1$ if $p_1 f_1(\mathbf{x}) > p_2 f_2(\mathbf{x})$, and $y = 1$ if $p_1 f_1(\mathbf{x}) < p_2 f_2(\mathbf{x})$.

Consequently, the continuum solution space is reduced to the space of two solutions. It is seen that (6.13) and (6.15) coincide when

$$1 - d(y) = \frac{\mathrm{d}l_1(y)}{\mathrm{d}y}$$

$$1 + d(y) = \frac{\mathrm{d}l_2(y)}{\mathrm{d}y}$$

These expressions allow one to introduce an additional physical interpretation of function 1 – $d(y)$.

Minimization of the average risk function (6.14) under limitations (6.7), (6.8) and condition

$$\int_Y \overbrace{\int_X \dots \int}^{N} G(\mathbf{x}, y) g_1(\mathbf{x}, y) \mathrm{d}\mathbf{x}\, \mathrm{d}y = \overbrace{\int_X \dots \int}^{N} g_1[\mathbf{x} P(\mathbf{x})] \mathrm{d}\mathbf{x} = 0 \tag{6.16}$$

where

$$g_1(\mathbf{x}, y) = p_1 f_1(\mathbf{x}) l_1(y) - p_2 f_2(\mathbf{x}) l_2(y)$$

gives the following equation for the neural network optimal model:

$$(1+\lambda) p_1 f_1(\mathbf{x}) \frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\bigg|_{y=P(\mathbf{x})} + (1+\lambda) p_2 f_2(\mathbf{x}) \frac{\mathrm{d}l_2(y)}{\mathrm{d}y}\bigg|_{y=P(\mathbf{x})} = 0 \tag{6.17}$$

The Langrangian multiplier λ is determined by (6.16), (6.17).

The limitation in the form of a given average risk function component has the form

$$\int_Y \overbrace{\int_X \dots \int}^{N} G(\mathbf{x}, y) [p_1 f_1(\mathbf{x}) l_1(y)] \mathrm{d}\mathbf{x}\, \mathrm{d}y = \alpha \tag{6.18}$$

If one were to denote

$$g_2(\mathbf{x}, y, \lambda) = g_2(\mathbf{x}, y) + \lambda p_1 f_1(\mathbf{x}) l_1(y) = (1+\lambda) p_1 f_1(\mathbf{x}) l_1(y) + p_2 f_2(\mathbf{x}) l_2(y)$$

then the expression for the neural network optimal model is

$$\frac{\partial g_2(\mathbf{x}, y, \lambda)\big|_{y=P(\mathbf{x})}}{\partial P(\mathbf{x})}$$

or in the other form,

$$(1+\lambda) p_1 f_1(\mathbf{x}) \frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\bigg|_{y=P(\mathbf{x})} + p_2 f_2(\mathbf{x}) \frac{\mathrm{d}l_2(y)}{\mathrm{d}y}\bigg|_{y=P(\mathbf{x})} = 0$$

The multiplier λ is determined by (6.14), (6.18).

Neural network of type 6. The pattern recognition system for K pattern classes and a continuum of solution under the average risk function minimum criterion has the following neural network optimal model:

$$\sum_{k=1}^{K} p_k f_k(\mathbf{x}) \left. \frac{\mathrm{d}l_k(y)}{\mathrm{d}y} \right|_{y=P(\mathbf{x})} = 0$$

where $P(\mathbf{x}) = y$ is the neural network optimal model.

Neural network of type 7. This is the recognition system for K pattern classes with two solutions. In the case of the average risk function minimum criterion, instead of matrix of loss coefficients that emerge at considering the i-th class patterns as belonging to the j-th class and having the form (p.1)

$$L = \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix}$$

one must introduce matrix

$$L = \begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \\ \cdots & \cdots \\ l_{k1} & l_{k2} \\ \cdots & \cdots \\ l_{K1} & l_{K2} \end{bmatrix}$$

The latter matrix is the matrix of loss coefficients that emerge when considering the k-th class patterns ($k = 1, \ldots, K$) as belonging to the regions of the multidimensional feature space corresponding to the first and second solutions. The expression for the conditional risk function is

$$r_k = l_{k1} \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})<0} f_k(\mathbf{x})\mathrm{d}\mathbf{x} + l_{k2} \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})>0} f_k(\mathbf{x})\mathrm{d}\mathbf{x}$$

The average risk function is the conditional risk function averaged across all the classes

$$R = \sum_{k=1}^{K} p_k r_k = \sum_{k=1}^{K} p_k l_{k1} \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})<0} f_k(\mathbf{x})\mathrm{d}\mathbf{x} + \sum_{k=1}^{K} p_k l_{k2} \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})>0} f_k(\mathbf{x})\mathrm{d}\mathbf{x}$$

Taking into account that

$$\overbrace{\int \cdots \int}^{N}_{-\infty} f_k(\mathbf{x})\mathrm{d}\mathbf{x} = \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})>0} f_k(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})<0} f_k(\mathbf{x})\mathrm{d}\mathbf{x}$$

the final expression for the average risk function is

$$R=\sum_{k=1}^{K} p_k l_{k2}+\overbrace{\int\ldots\int}^{N}_{S(\mathbf{x})<0}\left[\sum_{k=1}^{K} p_k(l_{k1}-l_{k2})\right] f_k(\mathbf{x})\mathrm{d}\mathbf{x}$$

It can be shown that minimum R is achieved at

$$S(\mathbf{x})=\sum_{k=1}^{K} p_k\left(l_{k1}-k_{k2}\right) f_k(\mathbf{x})$$

This is the equation for the optimal divisional surface that determines the neural network optimal model.

Neural network of type 8. The optimization of the neural network with a pattern class continuum and two solutions by the minimum risk function criterion requires the introduction of matrix (row-vector) $L=[l_1(\varepsilon), l_2(\varepsilon)]$ of the loss function corresponding to the first and second solutions. The conditional risk function is the risk function that makes a decision about the neural network input patterns belonging to the assemblage with distribution $f'(\mathbf{x}/\varepsilon)$. Typical loss functions $l_1(\varepsilon)$ and $l_2(\varepsilon)$ are shown in Fig. 6.11. The expression for the conditional risk function is

$$r(\varepsilon)=l_1(\varepsilon)\overbrace{\int\ldots\int}^{N}_{S(\mathbf{x})<0} f'(\mathbf{x}/\varepsilon)\mathrm{d}\mathbf{x}+l_2(\varepsilon)\overbrace{\int\ldots\int}^{N}_{S(\mathbf{x})>0} f'(\mathbf{x}/\varepsilon)\mathrm{d}\mathbf{x}$$

The equation $S(\mathbf{x}) = 0$ is the equation for the divisional surface in the multidimensional feature space.

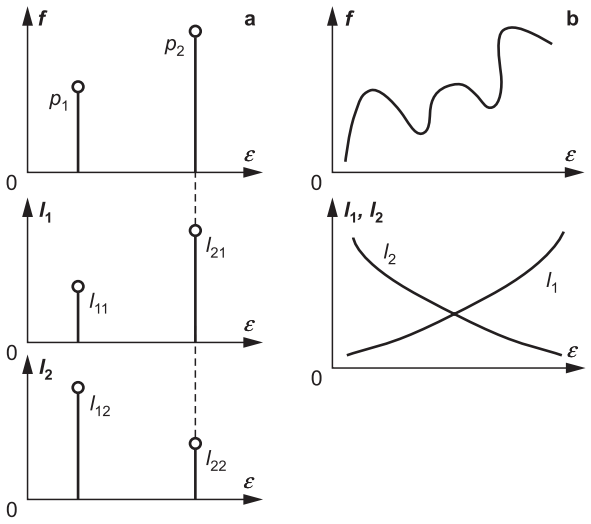


Fig. 6.11.
Loss functions for the case with a class continuum and two solutions: **a** two decisions; **b** class continuum

The average risk function is obtained by averaging across all the ε values

$$R = \int_{-\infty}^{\infty} r(\varepsilon) f_{\varepsilon}(\varepsilon) \mathrm{d}\varepsilon$$

$$= \int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon) \times \left[l_1(\varepsilon) \overbrace{\int_{S(\mathbf{x})<0} \cdots \int}^{N} f'(\mathbf{x}/\varepsilon) \mathrm{d}\mathbf{x} + l_2(\varepsilon) \overbrace{\int_{S(\mathbf{x})>0} \cdots \int}^{N} f'(\mathbf{x}/\varepsilon) \mathrm{d}\mathbf{x} \right] \mathrm{d}\varepsilon$$

The final expression after transformation is

$$R = \int_{-\infty}^{\infty} \left\{ l_2(\varepsilon) f_{\mathrm{g}}(\varepsilon) + f_{\mathrm{g}}(\varepsilon) [l_1(\varepsilon) - l_2(\varepsilon)] \overbrace{\int_{S(\mathbf{x})<0} \cdots \int}^{N} f'(\mathbf{x}/\varepsilon) \mathrm{d}\mathbf{x} \right\} \mathrm{d}\varepsilon$$

Or in the other form,

$$R = \int_{-\infty}^{\infty} l_2(\varepsilon) f_{\varepsilon}(\varepsilon) \mathrm{d}\varepsilon + \overbrace{\int_{S(\mathbf{x})<0} \cdots \int}^{N} \times \left\{ \int_{-\infty}^{\infty} [l_1(\varepsilon) - l_2(\varepsilon)] f_{\varepsilon}(\varepsilon) f'(\mathbf{x}/\varepsilon) \mathrm{d}\varepsilon \right\} \mathrm{d}\mathbf{x}$$

Consequently, the minimum is achieved at

$$S(\mathbf{x}) = \int_{-\infty}^{\infty} [l_1(\varepsilon) - l_2(\varepsilon)] f_{\varepsilon}(\varepsilon) f'(\mathbf{x}/\varepsilon) \mathrm{d}\varepsilon$$

Neural network of type 10. The neural network creates some divisional surface in the multidimensional feature space in the case of a pattern class continuum and K_{p} solutions.

The optimization by the minimum risk function criterion requires the introduction of matrix (row-vector)

$$L = \left[l_1(\varepsilon), \ldots, l_{K_{\mathrm{p}}}(\varepsilon) \right]$$

for the function of losses emerging due to consideration of patterns related objectively to the law $f'(\mathbf{x}/\varepsilon)$ as belonging to the regions of the multidimensional feature space corresponding to the first, second, K_{p}-th neural network solutions. The conditional risk function is

$$r(\varepsilon) = \sum_{k_{\mathrm{p}}=1}^{K_{\mathrm{p}}} l_{k_{\mathrm{p}}}(\varepsilon) \overbrace{\int_{S^{k_{\mathrm{p}}}(\mathbf{x})>0} \cdots \int}^{N} f'(\mathbf{x}/\varepsilon) \mathrm{d}\mathbf{x}$$

The average risk function after averaging $r(\varepsilon)$ is

$$R = \int_{-\infty}^{\infty} r(\varepsilon) f_{\varepsilon}(\varepsilon) \mathrm{d}\varepsilon = \int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon) \sum_{k_{\mathrm{p}}=1}^{K_{\mathrm{p}}} l_{k_{\mathrm{p}}}(\varepsilon) \overbrace{\int_{S^{k_{\mathrm{p}}}(\mathbf{x})>0} \cdots \int}^{N} f'(\mathbf{x}/\varepsilon) \mathrm{d}\mathbf{x} \, \mathrm{d}\varepsilon$$

Consequently, the optimal neural network model is determined by the system of inequalities:

$$S^{(k_p)}(\mathbf{x}) = \int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon)\left[l_{k_p''}(\varepsilon) - l_{k_p}(\varepsilon)\right] f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\varepsilon < 0 \quad , \quad k_p'' = 1,\ldots,K_p$$

Neural network of type 11. This is the system of continuum pattern class recognition with a solution continuum. The minimum average risk function optimization criterion requires not the matrix

$$L = \left[l_1(\varepsilon),\ldots,l_{k_p}(\varepsilon)\right]$$

as in the previous case, but the function of losses $l(y,\varepsilon)$ emerging due to consideration of the neural network input patterns related to the assemblage with distribution $f'(\mathbf{x}/\varepsilon)$. The conditional risk function is

$$r(\varepsilon) = \int_{Y} l(y,\varepsilon) \overbrace{\int_{X} \ldots \int}^{N} G(y,\mathbf{x}) f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y$$

The average risk function after averaging $r(\varepsilon)$ is

$$R = \int_{-\infty}^{\infty} r(\varepsilon) f_{\varepsilon}(\varepsilon)\,\mathrm{d}\varepsilon = \int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon) \int_{Y} l(y,\varepsilon) \overbrace{\int_{X} \ldots \int}^{N} G(y,\mathbf{x}) \times f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y\,\mathrm{d}\varepsilon$$

$$= \int_{Y} \overbrace{\int_{X} \ldots \int}^{N} G(y,\mathbf{x}) \times \left[\int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon) l(y,\varepsilon) f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\varepsilon\right] \mathrm{d}\mathbf{x}\,\mathrm{d}y$$

Let us designate

$$g_3(\mathbf{x},y) = \int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon) l(y,\varepsilon) f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\varepsilon$$

Then the expression for the average risk function is

$$R = \int_{Y} \overbrace{\int_{X} \ldots \int}^{N} G(y,\mathbf{x}) g_3(\mathbf{x},y)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y$$

Taking into account aforementioned G(**x**,*y*) properties, one obtains

$$R = \overbrace{\int_{X} \ldots \int}^{N} g_3[\mathbf{x}, P(\mathbf{x})]\,\mathrm{d}\mathbf{x}$$

where $y = P(\mathbf{x})$ is the optimal neural network model. The optimization solution gives the following condition:

$$\left.\frac{\partial g_3(\mathbf{x}, y)}{\partial y}\right|_{y=P(\mathbf{x})} = 0$$

and for particular form of $g_3(\mathbf{x},y)$,

$$\int_{-\infty}^{\infty} f_{\varepsilon}(\varepsilon) f'(\mathbf{x}/\varepsilon)\left[\frac{\partial}{\partial y} l(y,\varepsilon)\bigg|_{y=P(\mathbf{x})}\right] \mathrm{d}\varepsilon = 0$$

This is the most general expression for the neural network optimal model that takes into account each of the aforementioned cases.

6.3 Optimal Neural Network Model for Multidimensional Signals $\boldsymbol{\varepsilon}(n)$ and $y(n)$

The expression for the conditional risk function has the following form:

$$r(\varepsilon) = \overbrace{\int_{Y}\cdots\int}^{N} l(y,\varepsilon) \overbrace{\int_{X}\cdots\int}^{N} G(y,\mathbf{x}) f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y$$

Then the average risk function is

$$R = \overbrace{\int_{E}\cdots\int}^{N^*} r(\varepsilon) f_{\varepsilon}(\varepsilon)\,\mathrm{d}\varepsilon = \overbrace{\int_{Y}\cdots\int}^{N^*} \overbrace{\int_{X}\cdots\int}^{N} G(y,\mathbf{x}) \left[\overbrace{\int_{E}\cdots\int}^{N^*} f_{\varepsilon}(\varepsilon) l(y,\varepsilon) f'(\mathbf{x}/\varepsilon)\,\mathrm{d}\varepsilon\right] \mathrm{d}\mathbf{x}\,\mathrm{d}y$$

and after the introduction of additional designations,

$$R = \overbrace{\int_{Y}\cdots\int}^{N} \overbrace{\int_{X}\cdots\int}^{N} G(y,\mathbf{x}) g(y,\mathbf{x})\,\mathrm{d}\mathbf{x}\,\mathrm{d}y$$

As it is mentioned above, E is the neural network supervisor instruction space; and N^* is dimensionality of E and neural network output signal. If $N^* = 1$, then the neural network has K_{p} solutions, and function $G(\mathbf{x}, y)$ has the form

$$G\left(\mathbf{x}, K_{\mathrm{p}}\right) = \begin{cases} 1\ , & \text{if} \quad x \in S^{K_{\mathrm{p}}}(\mathbf{x}) > 0 \\ 0\ , & \text{if} \quad x \notin S^{K_{\mathrm{p}}}(\mathbf{x}) > 0 \end{cases}$$

and for the solution continuum,

$$G(\mathbf{x},y)=\begin{cases}1 \ , & \text{if} \quad y=P(\mathbf{x})\\ 0 \ , & \text{if} \quad y\neq P(\mathbf{x})\end{cases}$$

The transformation described by the system in the case of multidimensional E and Y can be written in the form

$$y(n)=\mathbf{P}[\mathbf{x}(n)] \ \text{ or } \ \begin{bmatrix} y_1(n)\\ \cdot\\ \cdot\\ \cdot\\ y_{N^*}(n)\end{bmatrix}=\begin{bmatrix} P_1(\mathbf{x})\\ \cdot\\ \cdot\\ \cdot\\ P_{N^*}(\mathbf{x})\end{bmatrix}.$$

If $N^*=\text{const.}$ and the neural network has a discrete number of the output signal (K_p solutions), then function $G(\mathbf{x},y)$ has the form

$$G\left(\mathbf{x},k_{1\text{p}},\ldots,k_{N^*\text{p}}\right)=\begin{cases}1 \ , & \text{if} \quad x\in S^{\left(k_{1\text{p}},\ldots,k_{N^*\text{p}}\right)}(\mathbf{x})>0\\ 0 \ , & \text{if} \quad x\notin S^{\left(k_{1\text{p}},\ldots,k_{N^*\text{p}}\right)}(\mathbf{x})>0\end{cases}$$

and for the solution continuum, respectively

$$G\left(\mathbf{x},y_1,\ldots,y_{N^*}\right)=\begin{cases}1 \ , & \text{if} \quad y=\mathbf{P}[\mathbf{x}(n)]\\ 0 \ , & \text{if} \quad y\neq\mathbf{P}[\mathbf{x}(n)]\end{cases}$$

Consequently, the expression for the average risk function is

$$R=\overbrace{\int_X\ldots\int}^{N} g[\mathbf{P}(\mathbf{x}),\mathbf{x}]\mathrm{d}\mathbf{x}=\overbrace{\int_X\ldots\int}^{N}\overbrace{\int_E\ldots\int}^{N^*} f_\varepsilon(\varepsilon)l[\mathbf{P}(\mathbf{x}),\varepsilon]f'(\mathbf{x}/\varepsilon)\mathrm{d}\varepsilon\,\mathrm{d}\mathbf{x}$$

The neural network optimal model is determined by the expression

$$\overbrace{\int_E\ldots\int}^{N^*} f_\varepsilon(\varepsilon)f'(\mathbf{x}/\varepsilon)\left[\frac{\partial}{\partial y}l(y,\varepsilon)\right]_{y=\mathbf{P}(\mathbf{x})}\mathrm{d}\varepsilon=0$$

where derivative

$$\frac{\partial}{\partial y}l\left(x_y,\varepsilon\right)$$

is the function of two variables y and ε.

Let each of the N^* output channels have K_0 amplitude gradations. Then the expression for the conditional risk function has the form

$$r(k_1,\ldots,k_{N^*}) = \sum_{k_{1\mathrm{p}}=1}^{K} \ldots \sum_{k_{N^*\mathrm{p}}=1}^{K} l(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}},k_1,\ldots,k_{N^*}) \times \overbrace{\int\ldots\int_{S^{(k_{1\mathrm{p}},\ldots k_{N^*\mathrm{p}})}(\mathbf{x})>0}}^{N} f'(\mathrm{x}/\varepsilon = k_1,\ldots,k_{N^*})\mathrm{d}\mathbf{x}$$

and the average risk function is

$$\begin{aligned} R &= \sum_{k_1=1}^{K} \ldots \sum_{k_{N^*}=1}^{K} r(k_1,\ldots,k_{N^*}) f_{\varepsilon}(k_1,\ldots,k_{N^*}) \\ &= \sum_{k_{1\mathrm{p}}=1}^{K} \ldots \sum_{k_{N^*\mathrm{p}}=1}^{K} \sum_{k_1=1}^{K} \ldots \sum_{k_{N^*}=1}^{K} l(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}},k_1,\ldots,k_{N^*}) \\ &\quad \times f_{\mathrm{g}}(k_1,\ldots,k_{N^*}) \overbrace{\int\ldots\int_{S^{(k_{1\mathrm{p}},\ldots k_{N^*\mathrm{p}})}(\mathbf{x})>0}}^{N} f'(\mathrm{x}/\varepsilon = k_1,\ldots,k_{N^*})\mathrm{d}\mathbf{x} \end{aligned}$$

After introducing additional designations

$$R = \sum_{k_{1\mathrm{p}}=1}^{K} \ldots \sum_{k_{N^*\mathrm{p}}=1}^{K} \overbrace{\int\ldots\int_{S^{(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}})}(\mathbf{x})>0}}^{N} g(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}},\mathbf{x})\mathrm{d}\mathbf{x}$$

where

$$g(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}},\mathbf{x}) = \sum_{k=1}^{K} \ldots \sum_{k_{N^*}=1}^{K} l(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}},k_1,\ldots,k_{N^*}) f_{\varepsilon}(k_1,\ldots,k_{N^*}) \times f'(\mathrm{x}/\varepsilon = k_1,\ldots,k_{N^*})$$

the result of the average risk function optimization in this case has the form

$$S^{(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}})}(\mathbf{x}) = g(k''_{1\mathrm{p}},\ldots,k''_{N^*\mathrm{p}},\mathbf{x}) - g(k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}},\mathbf{x}) > 0$$

$$(k''_{1\mathrm{p}},\ldots,k''_{N^*\mathrm{p}}) = \underbrace{(0\ldots0)}_{N^*},\ldots,\underbrace{(K,\ldots,K)}_{N^*}$$

i.e., K^{N^*} combinations exist.

It is possible to consider the case $K = 2$ as the most simple one for the implementation.

6.4 A Priori Information about the Input Signal in the Self-Learning Mode

The self-learning mode differs from the learning one because of the absence of information in the form of supervisor instructions about patterns belonging to a particular class. Consequently, this information must be represented in the neural network a priori. The following limitation for class determination in the self-learning mode is reasonable: only one mode of the input signal $\mathbf{x}(n)$ distribution density must correspond to each pattern class.

The a priori information about the input signal significantly influences the self-learning problem solving methods. This information can be divided into three main parts.

1. A priori information about the number of classes, i.e., about the number of input signal distribution density modes. The neural network input signal distribution can be represented according to the a priori information in the form

$$f(\mathbf{x}) = \sum_{k=1}^{K} p_k f_k(\mathbf{x}) \tag{6.19}$$

   where $\mathbf{x}(n)$ is the input signal; $f(\mathbf{x})$ is the input signal distribution density; $f_k(\mathbf{x})$ is the patterns distribution density in the k-th class; p_k is the probability of the k-th class pattern emergence; and K is the number of classes.
2. A priori information about the form of patterns distribution density in each class.
3. A priori information about probabilities p_k.

A priori information about the number of classes K (about the number of input signal distribution density modes) can be of three types ordered by the decrease of the a priori information: the number of classes (modes) is known exactly; the number of classes (modes) does not exceed some given $K_{\max}$; and the number of classes (modes) is not known.

The self-learning solution algorithm for the given number of classes must be developed in the first case. The self-learning algorithm optimal for the maximum number of classes and optimal for the smaller number of classes must be developed in the second case. The self-learning algorithm allows one to develop only a qualitative solution for the gradually increased number $K_{\max}$ in the third case. The termination criterion must be introduced in the latter case. The absence of the self-learning quality improvement at the $K_{\max}$ increase or the excessive algorithm complexity can be such a termination criterion.

A priori information about the form of pattern distribution density in each class can be of three types ordered by the decrease of the a priori information: the distribution form is known exactly; the distribution form is not known but some distribution approximation can be accepted; and the distribution form is not known.

The neural network optimal model implementation methods are dependent upon the a priori information quantity.

A priori information about probability of the k-th class pattern emergence. The a priori information for the representation (6.19) can be the following: coefficients p_k are the same for all classes; coefficients are different for all classes but unknown.

The first case does not impose any limitations upon the self-learning problem solving methods. The second case results in the self-learning process complexity due to the necessity to determine coefficients p_k in addition to determining distribution parameters for each subclass in the adjustment procedure.

6.5 About Neural Network Primary Optimization Criteria in the Self-Learning Mode

The primary optimization criterion also represents some additional information embedded in the neural network a priori. This criterion determines the quality of the recognition system that must be achieved in the self-learning mode.

It is possible to use the primary optimization criterion in all the aforementioned cases (the known pattern distribution, pattern distribution approximation, the unknown pattern distribution). Hence, the divisional surface is calculated according to the following expression:

$$\frac{\partial f(x)}{\partial x} = 0 \quad \text{under constraint} \quad \frac{\mathrm{d}^2 f(x)}{\mathrm{d}x^2} > 0$$

The solution of this equation corresponds to the threshold h_1 (Fig. 6.12).

The following criterion can be used when pattern distribution across classes can be determined or approximated:

$$p_1 f_1(x) = p_2 f_2(x) \tag{6.20}$$

The solution of this equation corresponds to the threshold h_2 (Fig. 6.12).

The use of the primary optimization criterion (6.20) in the self-learning mode corresponds to the idea of self-learning human in the case of two features and two classes (Fig. 6.13). The goal is to put a divisional surface through the places with the presence of minimum patterns.

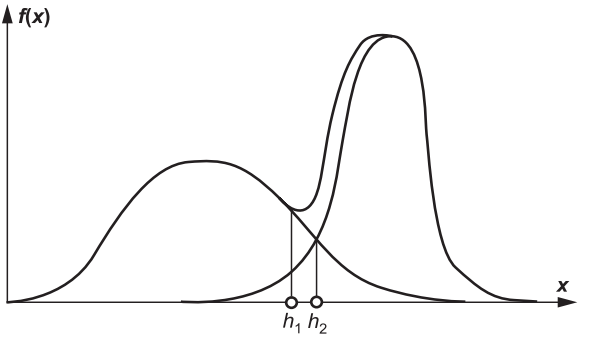


Fig. 6.12.
The introduction of primary optimization criteria for self-learning neural networks

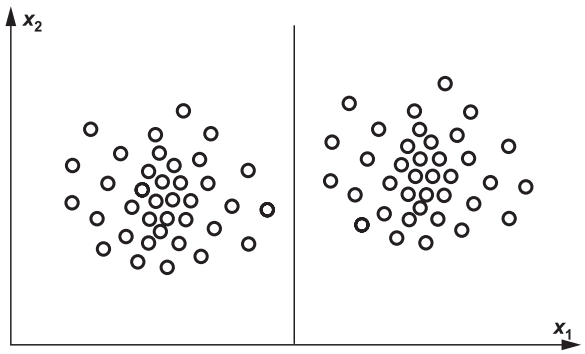


Fig. 6.13.
Illustration of criterion (6.20)

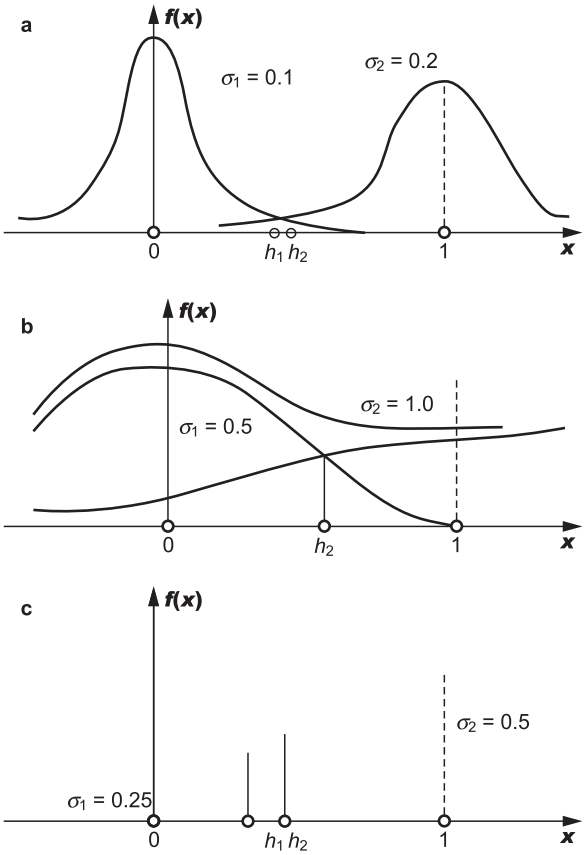


Fig. 6.14.
Comparison between primary optimization criteria in the self-learning mode

It can be shown that the optimal solutions according to the aforementioned primary optimization criteria in the self-learning mode are different. The particular case shown in Fig. 6.14 illustrates additional features of these criteria. Three possibilities can be mentioned:

1. Classes are easily selected, i.e., the crosscut is small. In this case, σ_1 and σ_2 are much smaller than the distance between centers of classes. The optimal thresholds h_1 and h_2 correspond to the first and second primary optimization criteria and are close (Fig. 6.14a);
2. The crosscut of classes is so large that the first criterion for the threshold h_1 is invalid. One of the parameters is larger than half of the distance between the centers of the classes, and the second one is comparable with it (Fig. 6.14b, $\sigma_2 = 1 > 0.5$);
3. The crosscut of classes is large, and thresholds h_1 and h_2 greatly differ. Parameters σ_1 and σ_2 are of the same order with half of the distance between the centers of the classes (Fig. 6.14c).

The obtained results have the following explanation: in the cases (1) and (3) the input signal distribution $f(x)$ is two-modal, and in case (2) it is one-modal with unclear qualitative sense for division of one hump of curve into two classes.

This is the reason for introducing limitations related to the modal characteristics of distribution density into the definition of classes in the problem of pattern recognition. The representation of $f(x)$ as a multi-modal function allows one to use a special average risk function as the primary optimization criterion in the self-learning mode.

6.6 Optimal Neural Network Models in the Self-Learning Mode and Arbitrary Teacher Qualification

Let us suppose that patterns are grouped around some unknown centers of $\mathbf{b}_{k_\mathrm{p}}$ classes. If the function of the distance between patterns and the k-th class is

$$\rho\left(\mathbf{x}, \mathbf{b}_{k_\mathrm{p}}\right) = \left\|\mathbf{x} - \mathbf{b}_{k_\mathrm{p}}\right\|^2$$

then the average risk function of $\mathbf{x}$ belonging to the region of the k_p-th solution can be represented in the form

$$r_{k_\mathrm{p}} = \int\limits_{S^{\left(k_\mathrm{p}\right)}(\mathbf{x})>0} \left\|\mathbf{x} - \mathbf{b}_{k_\mathrm{p}}\right\|^2 f(\mathbf{x})\mathrm{d}\mathbf{x}$$

where $\|\cdot\|$ is the norm of a vector. The average risk function is

$$R = \sum_{k_\mathrm{p}=1}^{K_\mathrm{p}} \int\limits_{S^{\left(k_\mathrm{p}\right)}(\mathbf{x})>0} \left\|\mathbf{x} - \mathbf{b}_{k_\mathrm{p}}\right\|^2 f(\mathbf{x})\mathrm{d}\mathbf{x}$$

The region of the k_p-th solution ($k_\mathrm{p} = 1, \ldots, K$) with minimum R is determined in this case by the following system of inequalities:

$$S^{\left(k_\mathrm{p}\right)}(\mathbf{x}) = \left\|\mathbf{x} - \mathbf{b}_{k'_\mathrm{p}}\right\|^2 - \left\|\mathbf{x} - \mathbf{b}_{k_\mathrm{p}}\right\|^2 > 0 \quad , \quad k'_\mathrm{p} \neq k_\mathrm{p} = 1, \ldots, K_\mathrm{p} \tag{6.21}$$

The equation for coordinates of class centers with a minimum R is

$$\mathbf{b}_{k_\mathrm{p}i} = \frac{\int\limits_{S^{(k_\mathrm{p})}(\mathbf{x})>0} \mathbf{x}_i f(\mathbf{x})\mathrm{d}\mathbf{x}_i}{\int\limits_{S^{(k_\mathrm{p})}(\mathbf{x})>0} f(\mathbf{x})\mathrm{d}\mathbf{x}} \quad , \quad i=1,\ldots,N \quad ; \quad k_\mathrm{p}=1,\ldots,K_\mathrm{p} \tag{6.22}$$

The systems (6.21), (6.22) determine the optimal neural network model in the self-learning mode. The loss function

$$\rho\left(\mathbf{x},\mathbf{b}_{k_\mathrm{p}}\right)=\left\|\mathbf{x}-\mathbf{b}_{k_\mathrm{p}}\right\|^2$$

is a rather rough approximation of distribution inside the class. A more precise approximation can be achieved by complication of the optimal model at the expense of the loss function complication in the following way:

$$\rho\left(\mathbf{x},\mathbf{b}_{k_\mathrm{p}}\right)=\left\|\frac{\mathbf{x}-\mathbf{b}_{k_\mathrm{p}}}{\sigma_{k_\mathrm{p}}}\right\|^2$$

or by using a more complex function $\rho(\mathbf{x},\mathbf{b}_{k_\mathrm{p}}) = ||\mathbf{x}–\mathbf{b}_{k_\mathrm{p}}||^2$.

Similar to the case of the learning mode at the transfer to the solution continuum, the loss function $\rho[\mathbf{x},\mathbf{b}(y)]$ is introduced in the case of the self-learning mode. Here $\mathbf{b}(y)$ is either the final or intermediate result of the neural network synthesis. The average risk function has the following form in the case of a discrete set of solutions:

$$R=\sum_{k_\mathrm{p}=1}^{K_\mathrm{p}}\int_X \rho\left(\mathbf{x},\mathbf{b}_{k_\mathrm{p}}\right)G\left(\mathbf{x},k_\mathrm{p}\right)f(\mathbf{x})\mathrm{d}\mathbf{x}$$

where

$$G\left(\mathbf{x},k_\mathrm{p}\right)=\begin{cases}1 \ , & \text{if} \quad x\in S^{(k_\mathrm{p})}(\mathbf{x})>0\\ 0 \ , & \text{if} \quad x\notin S^{(k_\mathrm{p})}(\mathbf{x})>0\end{cases}$$

In the case of a solution continuum one obtains

$$R=\int_Y\int_X \rho[\mathbf{x},\mathbf{b}(y)]G\left(\mathbf{x},y\right)f(\mathbf{x})\mathrm{d}\mathbf{x}\,\mathrm{d}y$$

where

$$G\left(\mathbf{x},y\right)=\begin{cases}1 \ , & \text{if} \quad y\in P(\mathbf{x})>0\\ 0 \ , & \text{if} \quad y\notin P(\mathbf{x})>0\end{cases}$$

or in another form,

$$R = \int_X \rho\left[\mathbf{x}, \mathbf{b}\left[P(\mathbf{x})\right]\right] f(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

The expression for the optimal neural network model is obtained by derivation of R by $y = P(\mathbf{x})$ similar to the case of the learning mode:

$$f(\mathbf{x})\frac{\partial}{\partial y}\rho\left[\mathbf{x}, \mathbf{b}(y)\right]\Big|_{y=P(\mathbf{x})} = 0 \quad , \quad f(\mathbf{x})\frac{\partial}{\partial b_i}\rho\left[\mathbf{x}, \mathbf{b}(y)\right] = 0 \quad , \quad i = 1, \ldots, N$$

with additional Silvester conditions for the matrix of mixed derivatives.

Let us consider the neural network optimal model with $K_\mathrm{p} = K$ solutions under the arbitrary teacher qualification b.

One must define the loss function $l(\mathbf{x}, \mathbf{b}_{k_\mathrm{p}}, \mathbf{b}, l_{k_\mathrm{p}k})$ in such a way that in the learning mode, when $b = 1$ then $l = l_{k_\mathrm{p}k}$, in the self-learning mode, when $b = 0$ then $l = \rho(\mathbf{x}, \mathbf{b}_{k_\mathrm{p}})$, and when $b = -1$ then the primary optimization functional with the extreme inverse respective of the learning mode. Such a loss function can be written in the following form:

$$l\left(\mathbf{x}, \mathbf{b}_{k_\mathrm{p}}, \mathbf{b}, l_{k_\mathrm{p}k}\right) = b l_{k_\mathrm{p}k} + \left(1 - b^2\right)\rho\left(\mathbf{x}, \mathbf{b}_{k_\mathrm{p}}\right)$$

The expression for the average risk function is

$$R = \sum_{k_\mathrm{p}=1}^{K} \int_{S^{(k_\mathrm{p})}(\mathbf{x})>0} \left\{\left[\sum_{k=1}^{K} p_k f_k(\mathbf{x}) l_{k_\mathrm{p}k}\right] + (1 - b^2)\left\|\mathbf{x} - \mathbf{b}_{k\mathrm{p}}\right\|^2 f(\mathbf{x})\right\}\mathrm{d}\mathbf{x}$$

The optimal region of the k_p-th solution can be represented in the following form:

$$S^{(k_\mathrm{p})}(\mathbf{x}) = g_{k'_\mathrm{p}}(\mathbf{x}) - g_{k_\mathrm{p}}(\mathbf{x}) > 0 \quad , \quad k'_\mathrm{p} \neq k_\mathrm{p} = 1, \ldots, K_\mathrm{p}$$

The expression for optimal values $b_{k_\mathrm{p}i}$ is similar to (6.22).

It was assumed above that the teacher qualification in the neural network optimal model development is exactly known. The approximate (not exact) knowledge of the teacher qualification is usually observed, for example, in the field of medical diagnostic solution. In the case of $K = K_\mathrm{p}$ classes and neural network solutions, one obtains

$$R = \sum_{k_\mathrm{p}=1}^{K} \int_{S^{(k_\mathrm{p})}(\mathbf{x})>0} \left\{\sum_{k=1}^{K}\left[b_\mathrm{c} p_k l_{k_\mathrm{p}k} + (1 - b_\mathrm{c}^2)\rho(\mathbf{x}, \mathbf{b}_{k_\mathrm{p}})\right] f'(\mathbf{x}/\varepsilon = k)\right\}\mathrm{d}\mathbf{x}$$

The optimal neural network model determined by the system of K inequalities can be written in this case in the form

$$\sum_{k=1}^{K}\left\{b_c(p_k l_{kk_p} - p_k l_{kk'_p}) + (1-b_c^2)\left[\rho(\mathbf{x},\mathbf{b}_{k_p}) - \rho(\mathbf{x},\mathbf{b}_{k'_p})\right]\right\}\frac{\sum_{k'=1}^{K} p_{k'} f_{k'}(\mathbf{x}) a_{kk'}}{\sum_{k'=1}^{K} p_{k'} a_{kk'}} > 0 \quad (6.23)$$

where $k'_p = 1, \ldots, K$.

It is supposed in this case that the subjective teacher qualification does not depend upon the class number.

Consequently, it is seen from (6.23) that when $b_c = 1$ and $[a_{kk'}] = A_1$ (A_1 is a unit matrix), then one deals with the learning mode. In the case $b_c = 0$ and arbitrary values of $a_{k'k}$, one deals with the self-learning mode. In the general case $b_c = b$, the system is adjustable. If the teacher qualification is zero then the system is not adjustable in the learning mode.

All of this indicates a significance of the a priori information required for the neural network optimal model design. Sometimes, it is not necessary to have such information. The amount of the a priori information about the form of $f'(\mathbf{x}/\varepsilon)$ determines methods of the neural network optimal model's implementation represented in Table 6.1.

Literature

[6-1] Galushkin AI (1970) Multilayer pattern recognition systems. Moscow, MIEM, p. 167

[6-2] Galushkin AI (1972) The choice of the primary optimization criteria and the optimal model design for *K* pattern class recognition in the learning mode. Automated control and computer technology 10

[6-3] Galushkin AI, Zotov Yu L, Shikunov Yu A (1972) In-line processing of experimental information. Moscow, Energiya p 360

[6-4] Galushkin AI (1969) Methods of pattern recognition systems synthesis. MIEM proc., 6:133–172

[6-5] Galushkin AI (1970) Unified approach to the learning and self-learning problems for pattern recognition systems. MIEM proc. 6:104–120

[6-6] Victorov NV, Galushkin AI (1976) Design and investigation of pattern recognition systems under the arbitrary teacher qualification. Medical radio-electronics, pp 95–106

Chapter 7

Analysis of the Open-Loop Neural Networks

7.1 Distribution Laws of Analogous and Discrete Neural Network Errors

The initial data for the analysis of the open-loop neural networks are the given distribution density of the input signal and the structure of the open-loop neural network. The following open-loop neural network structures are usually considered: neurons with two, K_p, and a continuum of solutions, nonlinear and multilayer neural networks.

The goal of the open-loop neural network analysis is the investigation of expressions for distributions and moments of distributions of intermediate and output neural network signals. This chapter mainly concerns the analysis of distributions and moments of distributions for the neural network errors. The functionals of the secondary optimization are selected on the basis of the open-loop neural network analysis.

The functional of the secondary optimization is considered as a functional expressed through the distribution parameters of the current neural network signals and errors that are directly minimized by the multilayer neural network under the closed-cycle adjustment. The main problem is the creation of the secondary optimization functional corresponding to the given primary optimization criterion. The coincidence of the neural network parameters providing minimums of primary and secondary functionals is considered as a desired correspondence.

7.1.1 Neuron with Two Solutions

The transformation performed by the neuron with two solutions can be represented in the following form:

$$y(n) = \operatorname{sign} \sum_{i=0}^{N} a_i x_i(n) = \operatorname{sign} g(n) \tag{7.1}$$

The expressions for analogous and discrete neuron errors have the form

$$x_a(n) = \varepsilon(n) - g(n) \quad ; \quad x_g(n) = \varepsilon(n) - y(n) \tag{7.2}$$

The input signal distribution function with $K=2$ is (Chap. 5)

$$f(\mathbf{x},\varepsilon)=\begin{cases}\dfrac{1}{4}A_1f_1(\mathbf{x})+\dfrac{1}{4}A_2f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon=1\\ \dfrac{1}{4}B_1f_1(\mathbf{x})+\dfrac{1}{4}B_2f_2(\mathbf{x}) \ , & \text{if} \quad \varepsilon=-1\end{cases}$$

Here

$$\begin{aligned}A_1&=\left[2+(c_2-c_1)-b_1(c_1+c_2)\right]p_1\\ A_2&=\left[2+(c_2-c_1)+b_2(c_1+c_2)\right]p_2\\ B_1&=\left[2+(c_2-c_1)+b_1(c_1+c_2)\right]p_1\\ B_2&=\left[2+(c_2-c_1)-b_2(c_1+c_2)\right]p_2\end{aligned}$$

The analogous error distribution of the considered neural network is

$$\begin{aligned}f_a(x)=\frac{1}{4a_N}\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N-1}\Bigg[&A_1f_1\left(x_1,\ldots,x_{N-1},\frac{1-a_0-x_a}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\right)\\ &+A_2f_2\left(x_1,\ldots,x_{N-1},\frac{1-a_0-x_a}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\right)\\ &+B_1f_1\left(x_1,\ldots,x_{N-1},\frac{-1-a_0-x_a}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\right)\\ &+B_2f_2(x_1,\ldots,x_{N-1},\frac{-1+a_0-x_a}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\Bigg]\mathrm{d}x_{N-1},\ldots,\mathrm{d}x_1\end{aligned} \tag{7.3}$$

and the discrete error distribution is

$$f(\mathbf{x},\varepsilon)=\begin{cases}\dfrac{1}{4}\left[B_1(1-\Phi_1)+B_2(1-\Phi_2)\right] \ , & \text{if} \quad x_g=-2\\ \dfrac{1}{4}\left[A_1(1-\Phi_1)+B_1\Phi_1+A_2(1-\Phi_2)+B_2\Phi_2\right] \ , & \text{if} \quad x_g=0\\ \dfrac{1}{4}\left[A_1\Phi_1+A_2\Phi_2\right] \ , & \text{if} \quad x_g=2\end{cases} \tag{7.4}$$

Here

$$\Phi_k=\Phi_k\left(\frac{a_0}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)=\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N-1}\int_{-\frac{a_0}{a_N}-\sum_{i=1}^{N-1}\frac{a_i}{a_N}x_i}^{\infty}f_k(x_1,\ldots x_N)\mathrm{d}x_1,\ldots,\mathrm{d}x_N \ , \quad k=1,2$$

Expressions for the r-th order distribution moments in the case of analogous and discrete errors of the considered neural network can be represented in the form

$$\alpha_{ra}=\frac{1}{4}\sum_{m=0}^{r}(-1)^{2r+m}C_r^m\sum_{i_1\ldots i_m=1}^{N}a_{i_1}\ldots a_{i_m}$$
$$\times\left\{\left[A_1(a_0+1)^{r-m}+B_1(a_0-1)^{r-m}\right|_{\gamma_{i_1,\ldots,i_m}}^{(1)}+\left[A_2(a_0+1)^{r-m}+B_2(a_0-1)^{r-m}\right|_{\gamma_{i_1,\ldots,i_m}}^{(2)}\right\} \quad (7.5)$$

$$\gamma_{i_1,\ldots,i_m}^{(k)}=\overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N}x_{i_1}\ldots x_{i_m}f_k(\mathbf{x})\mathrm{d}x$$

$$\alpha_{rg}=2^{r-2}\left[\left(A_1\Phi_1+A_2\Phi_2\right)+(-1)^r\left(B_1+B_2-B_1\Phi_1+B_2\Phi_2\right)\right] \quad (7.6)$$

In the particular case of $c_1=c_2=1$ and $b_1-b_2=b$, the distribution of the analogous error of the considered neural network has the form (7.3) with the substitution

$A_1=2(1-b)p_1$; $A_2=2(1+b)p_2$

$B_1=2(1+b)p_1$; $B_2=2(1-b)p_2$

The discrete error distribution has the form

$$f(\mathbf{x},\varepsilon)=\begin{cases}\frac{1}{2}\left[p_1(1+b)(1-\Phi_1)+p_2(1-b)(1-\Phi_2)\right], & \text{if} \quad x_g=-2\\ \frac{1}{2}\left[1+b(p_2-p_1)+2p_1b\Phi_1-2p_2b\Phi_2\right], & \text{if} \quad x_g=0\\ \frac{1}{2}\left[(1-b)p_1\Phi_1+(1+b)p_2\Phi_2\right], & \text{if} \quad x_g=2\end{cases}$$

The expression for the neural network discrete error moments of distribution has in the given case the following form:

$$\alpha_{rg}=2\left\{p_1\left[1+b(1-2\Phi_1)\right]+p_2\left[1-b(1-2\Phi_2)\right]\right\}$$

and separately for pattern assemblages of the first and second classes:

$$\alpha_{r1g}=2\left[p_1(1+b)(1-\Phi_1)+p_2(1-b)(1-\Phi_2)\right]$$

$$\alpha_{r2g}=2\left[p_1(1-b)\Phi_1+p_2(1+b)\Phi_2\right]$$

7.1.2
Neuron with a Solution Continuum

The transformation performed by the neuron with a solution continuum in the learning mode can be represented in the following form:

$$y(n)=F\left[\sum_{i=0}^{N}a_i x_i\right]=F\left[g(n)\right]$$

In the case of a neural network input class continuum,

$$f(\mathbf{x},\varepsilon)=f'(x_1,\ldots,x_N/\varepsilon)f_\varepsilon(\varepsilon)$$

The joint distribution for the signal $\varepsilon(n)$ and analogous output signal $g(n)$ has the form

$$f_{g\varepsilon}(g,\varepsilon)=\frac{1}{a_N}\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N-1}f'\left(x_1,\ldots,x_{N-1},\frac{g+a_0}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\middle/\varepsilon\right)f_\varepsilon(\varepsilon)\,\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1$$

The neural network analogous error distribution is

$$f_a(x_a)=\frac{1}{a_N}\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N-1}f'\left(x_1,\ldots,x_{N-1},\frac{-x_a+\varepsilon+a_0}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\middle/\varepsilon\right)f_\varepsilon(\varepsilon)\,\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1\,\mathrm{d}\varepsilon$$

Consequently, the expression for the r-th order moment of the analogous error is

$$\alpha_{ra}=\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N}\left(-\sum_{i=1}^{N}a_i y_i+\varepsilon+a_0\right)^r\times f'(\mathbf{y}/\varepsilon)f_\varepsilon(\varepsilon)\,\mathrm{d}\mathbf{y}\,\mathrm{d}\varepsilon \tag{7.7}$$

The neural network discrete error distribution is

$$f_{x_g}(x_g)=\frac{1}{a_N}\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N}f'\left(x_1,\ldots,x_{N-1},\frac{F^{-1}(\varepsilon-x_g)+a_0}{a_N}-\sum_{i=1}^{N-1}\frac{a_i}{a_N}x_i\middle/\varepsilon\right)\times f_\varepsilon(\varepsilon)\left|\frac{\mathrm{d}F^{-1}(\varepsilon-x_g)}{\mathrm{d}(\varepsilon-x_g)}\right|\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1\,\mathrm{d}\varepsilon \tag{7.7a}$$

Consequently,

$$\alpha_{rg}=\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N+1}\left[-F\left(-\sum_{i=1}^{N}a_i y_i-a_0\right)+\varepsilon\right]^r f'(\mathbf{y}/\varepsilon)f_\varepsilon(\varepsilon)\,\mathrm{d}\mathbf{y}\,\mathrm{d}\varepsilon \tag{7.8}$$

In the particular case of two pattern recognition learning, one obtains

$$
\begin{aligned}
f_{x_g}(x_g) = \frac{p_1}{a_N} \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N-1} \left| \frac{\mathrm{d}F^{-1}(-1-x_g)}{\mathrm{d}(-1-x_g)} \right| \\
\times f_1\left(x_1,\dots,x_{N-1}, \frac{F^{-1}(-1-x_g)+a_0}{a_N} - \sum_{i=1}^{N-1} x_i \frac{a_i}{a_N} \right) \\
\times \mathrm{d}x_{N-1}\dots\mathrm{d}x_1 + \frac{p_2}{a_N} \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N-1} \left| \frac{\mathrm{d}F^{-1}(1-x_g)}{\mathrm{d}(1-x_g)} \right| \\
\times f_2\left(x_1,\dots,x_{N-1}, \frac{F^{-1}(1-x_g)+a_0}{a_N} - \sum_{i=1}^{N-1} x_i \frac{a_i}{a_N} \right) \mathrm{d}x_{N-1}\dots\mathrm{d}x_1
\end{aligned}
\tag{7.8a}
$$

$$
\begin{aligned}
\alpha_{rg} = p_1 \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left[-F\left(-\sum_{i=1}^{N} a_i y_i - a_0 \right) - 1 \right]^r f_1(\mathbf{y})\mathrm{d}\mathbf{y} \\
+ p_2 \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left[-F\left(-\sum_{i=1}^{N} a_i y_i - a_0 \right) + 1 \right]^r f_2(\mathbf{y})\mathrm{d}\mathbf{y}
\end{aligned}
\tag{7.9}
$$

and separately for pattern assemblages of the first and second classes

$$
\left.
\begin{aligned}
\alpha_{r1g} = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left[-F\left(-\sum_{i=1}^{N} a_i y_i - a_0 \right) - 1 \right]^r f_1(\mathbf{y})\mathrm{d}\mathbf{y} \\
\alpha_{r2g} = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left[-F\left(-\sum_{i=1}^{N} a_i y_i - a_0 \right) + 1 \right]^r f_2(\mathbf{y})\mathrm{d}\mathbf{y}
\end{aligned}
\right\}
\tag{7.10}
$$

The latter expressions allow one to obtain the expressions for some particular cases:

$$
F(g) = \operatorname{sign} g \;\; ; \quad F(g) = \begin{cases} 1 \;, & \text{if} \quad g > \Delta a \\ 0 \;, & \text{if} \quad -\Delta a < g < \Delta a \\ -1 \;, & \text{if} \quad g < -\Delta a \end{cases}
$$

$$
F(g) = \begin{cases} 1 \;, & \text{if} \quad g > \Delta a \\ \dfrac{g}{\Delta a} \;, & \text{if} \quad -\Delta a < g < \Delta a \\ -1 \;, & \text{if} \quad g < -\Delta a \end{cases}
$$

7.1.3 Analysis of a Neuron with K_p Solutions

The joint probability distribution of the input signal for the recognition system of K pattern classes in the learning mode has the form

$$f(\mathbf{x},\varepsilon)=\begin{cases} p_1 f_1(\mathbf{x}) \ , & \text{if} \quad \varepsilon=1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ p_K f_K(\mathbf{x}) \ , & \text{if} \quad \varepsilon=K \end{cases}$$

In this case,

$$y(n)=k_{\mathrm{p}} \ , \quad \text{if} \quad a_{k_{\mathrm{p}}-1,k_{\mathrm{p}}}<g(n)<a_{k_{\mathrm{p}},k_{\mathrm{p}}+1} \ \left(a_{K,K+1}=\infty, a_{01}=-\infty\right)$$

For the assemblage of K-class patterns,

$$f_{ka}(x_a)=\frac{1}{a_N}\overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N-1} f_k\left(x_1,\ldots,x_{N-1},\frac{k+a_0-x_a}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\right)\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1$$

The analogous error distribution will have the following form:

$$f_a(x_a)=\sum_{k=1}^{K}p_k f_{ka}(x_a)=\sum_{k=1}^{K}\frac{p_k}{a_N}\overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N-1}$$

$$f_k\left(x_1,\ldots,x_{N-1},\frac{k+a_0-x_a}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}\right)\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1$$

The distribution of discrete output signal of the considered neuron type for the k-th class can be obtained in the following form:

$$\begin{aligned} f_k(y)=&\Phi_k\left(\frac{a_{k_{\mathrm{p}},k_{\mathrm{p}}+1}+a_0}{a_N};\frac{a_1}{a_N};\ldots;\frac{a_{N-1}}{a_N}\right) \\ &-\Phi_k\left(\frac{a_{k_{\mathrm{p}}-1,k_{\mathrm{p}}}+a_0}{a_N};\frac{a_1}{a_N};\ldots;\frac{a_{N-1}}{a_N}\right) \ , \quad \text{if} \quad Y=k_{\mathrm{p}} \end{aligned} \tag{7.10a}$$

Similar to the previous case, here

$$\Phi_k\left(\frac{a_{k_{\mathrm{p}}-1,k_{\mathrm{p}}}+a_0}{a_N};\frac{a_1}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)$$

$$=\overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N-1}\int_{-\infty}^{\frac{a_{k_{\mathrm{p}}-1,k_{\mathrm{p}}}+a_0}{a_N}-\sum_{i=1}^{N-1}x_i\frac{a_i}{a_N}} f_k(x_1,\ldots,x_N)\mathrm{d}x_1\ldots\mathrm{d}x_N$$

Consequently, the discrete error distribution is

$$f_{x_g}(x_g)=\sum_{k=1}^{K} p_k\left[\Phi_k\left(\frac{a_{k-k'',k-k''+1}+a_0}{a_N};\frac{a_1}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)\right.$$
$$\left.-\Phi_k\left(\frac{a_{k-k''-1,k-k''}+a_0}{a_N};\frac{a_1}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)\right] \quad \text{if} \quad x_g=k''$$

The expressions for the r-th order moments of distributions for analogous and discrete errors can be represented in the form

$$\alpha_{ra}=\sum_{k=1}^{K} p_k \sum_{m=0}^{r} C_r^m (a_0+k)^m (-1)^{r-m} \overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N}\left(\sum_{i=1}^{N} a_i y_i\right)^{r-m} f_k(\overline{x})\,\mathrm{d}\overline{x}$$

$$\alpha_{rg}=\sum_{k''=1-k}^{K}(k'')^r \sum_{k=1}^{K} p_k\left[\Phi_k\left(\frac{a_{k-k'',k-k''+1}+a_0}{a_N};\frac{a_1}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)\right.$$
$$\left.-\Phi_k\left(\frac{a_{k-k''-1,k-k''}+a_0}{a_N};\frac{a_1}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)\right]$$

The expression for the r-th moment of distribution for analogous error has the form

$$\alpha_{ra}=\frac{1}{a_N}\overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N} x_a^r \sum_{k=1}^{K} p_k f_k\left(x_1,\ldots,x_{N-1},\frac{k+a_0-x_a}{a_N}-\sum_{i=1}^{N-1} x_i\frac{a_i}{a_N}\right)\mathrm{d}x_a\,\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1$$

After the change of variables $x_1=y_1;\ldots\ x_{N-1}=\ y_{N-1}$,

$$\frac{k+a_0-x_a}{a_N}-\sum_{i=1}^{N-1} x_i\frac{a_i}{a_N}=y_N$$

one obtains

$$\alpha_{ra}=\sum_{k=1}^{K} p_k \overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N}\left[\sum_{i=1}^{N} y_i a_i+(r+a_0)\right]^r f_k(y_1,\ldots,y_N)\,\mathrm{d}y_N\ldots\mathrm{d}y_1$$

$$\alpha_{ra}=\sum_{k=1}^{K} p_k \overbrace{\int_{-\infty}^{\infty}\ldots\int}^{N-1}\sum_{m=0}^{r} C_r^m (a_0+k)^m (-1)^{r-m}\left(\sum_{i=1}^{N} a_i y_i\right)^{r-m} f_k(y)\,\mathrm{d}y$$

7.1.4
Analysis of a Pattern Recognition System with a Nonlinear Divisional Surface

It was shown in [7-4] that a pattern recognition system with a nonlinear divisional surface can be represented by the equivalent system consisting of an inertialess layer of nonlinear transformations and a neuron. It can be shown that if the nonlinear transformation layer forms vector components $(x_1,\ldots,x_N,\{z_{i_1,i_2}\},\ldots,\{z_{i_1,\ldots,i_r}\})$ from vector components $(x_1,\ldots,x_N)$, where $i_1,\ldots,i_r=1,\ldots,N$, and $z_{i_1,\ldots,i_r}=x_{i_1,\ldots,i_r}$, then the distribution density of this layer output signal can be represented by the distribution density $f(\mathbf{x})$ in the following way:

$$f'(x')=f'\left(\mathbf{x},\left\{z_{i_1,i_2}\right\},\ldots,\left\{z_{i_1,\ldots,i_r}\right\}\right)=\begin{cases} f(\mathbf{x}) \ , & \text{for all } i,k,(k=1,\ldots,r), \\ & \text{at which } z_{i_1,\ldots,i_k}=x_{i_1},\ldots,x_{i_k} \\ 0 \ , & \text{for all } i,k, \text{at which} \\ & z_{i_1,\ldots,i_k}\neq x_{i_1},\ldots,x_{i_k},(k=1,\ldots,r) \end{cases}$$

The expression for the second moment of distribution for the discrete error of the considered nonlinear system has the form

$$\alpha_{2g}=4\left[\Phi_2'p_2+p_1-\Phi_1'p_1\right]$$

where

$$\Phi_i'=\overbrace{\int\limits_{\substack{\infty \\ \cdots \\ S'(x')<0}}^{N^*}}\int f_i'(x')\mathrm{d}x' \quad \text{and} \quad S'(x')=-a_0+\sum_{i=1}^{N^*}a_ix_i'$$

It must be taken into account that the expression

$$S(x)=-a_0+\sum_{i=1}^{N}a_ix_i=0$$

determines a linear divisional surface in the initial feature space. Let us determine the change of divisional surface in the initial feature space under the increase of the order r of nonlinear transformation. In the case of the second order transformation,

$$\Phi_i'=\overbrace{\int\limits_{\substack{\infty \\ \cdots \\ S'(x')<0}}^{N^*}}\int\begin{Bmatrix} f_i(\mathbf{x}) & \text{if} & z_{i_1i_2}=x_{i_1i_2} \\ 0 & \text{if} & z_{i_1i_2}\neq x_{i_1i_2} \end{Bmatrix}\mathrm{d}x_1'\ldots\mathrm{d}x_{N^*}'$$

$$=\overbrace{\int\limits_{\substack{\infty \\ \cdots \\ S'(x')<0}}^{N^*}}\int f_1(\mathbf{x})\prod_{\substack{i_1=1 \\ i_2=1}}^{N}\delta(z_{i_1i_2}-x_{i_1i_2})\mathrm{d}x_1'\ldots\mathrm{d}x_{N^*}'$$

$$\Phi_i' = \overbrace{\int_{-a_0+\sum_{i=1}^{N} a_i x_i + \sum_{\substack{i_1=1\\ i_2=1}}^{N} a_{i_1 i_2} z_{i_1 i_2} < 0} \cdots \int}^{N^*} f_1(\mathbf{x}) \prod_{\substack{i_1=1\\ i_2=1}}^{N} \delta(z_{i_1 i_2} - x_{i_1 i_2}) \mathrm{d}x_1 \ldots \mathrm{d}x_N \prod_{\substack{i_1=1\\ i_2=1}}^{N} z_{i_1 i_2}$$

$$= \overbrace{\int_{-a_0+\sum_{i=1}^{N} a_i x_i + \sum_{\substack{i_1=1\\ i_2=1}}^{N} a_{i_1 i_2} x_{i_1} x_{i_2} < 0} \cdots \int}^{N^*} f_i(\mathbf{x}) \mathrm{d}\mathbf{x}$$

Consequently, if $r = 2$, then the equivalent divisional surface in the initial feature space will be the surface of the second order with coefficients determined in a unique manner by the coefficients of the output neuron with the input layer of the nonlinear transformations. In the r-th order transformation,

$$\alpha_{2g} = 2 \left[p_1 + \overbrace{\int_{-a_0+\sum_{k=1}^{r} \sum_{i_1,\ldots,i_k=1}^{N} a_{i_1 \ldots i_k} x_{i_1} \ldots x_{i_k} < 0} \cdots \int}^{N^*} [p_2 f_2(\mathbf{x}) - p_1 f_1(\mathbf{x})] \mathrm{d}\mathbf{x} \right]$$

This proves the equivalence (by the average risk function criterion) of representation of the pattern recognition system with nonlinear divisional surface in the form of nonlinear transformations units and a neuron.

7.2 Selection of the Secondary Optimization Functional

Let us consider the secondary optimization functional related to the moments of distribution of analogous and discrete errors for the neural network with two solutions. The general requirements for the neural network secondary optimization were mentioned in the introduction. The functional parameters that are required for the iteration search procedure must be sufficiently and simply measured and evaluated. This functional must have a simple form relative to the adjustable neural network coefficients. It must be minimal at the same adjustable neural network parameter values that provide an extremum of some primary optimization functional.

The analysis of expressions (7.5) and (7.6) for the moments of distribution activation function analogous and discrete error results in the following conclusion [7-2, 7-3]:

1. The odd-numbered moments of neural network analogous and discrete error distribution in the learning mode cannot be used as a secondary optimization functional. However, their absolute values can be used as such functionals.
2. The even-numbered moments of the aforementioned distributions can be used as the secondary optimization functional. In the case of discrete error, only consideration of the first and second moments order must be performed because the moments of higher order are proportional to α_{2g}.
3. The main goal of this chapter is the analysis of the primary optimization criterion, the a priori information about the input signal characteristics and the loss matrix corresponding to the minimization of the selected secondary optimization functional.
4. The analysis of the expressions for $|\alpha_{1a}|$ and α_{2a} in the case of a neuron shows that the minimization of these secondary optimization functionals is equivalent to the minimization of the average risk function under consideration of only the first order moments of distribution for pattern assemblages of different classes. It is considered that the emergence of a priori probability activation function patterns is the same for all classes, and the following restrictions upon the loss matrix coefficients takes place: $(l_{22} - l_{21}) = (l_{11} - l_{12})$.
5. The analysis of the expression for the absolute value of the first moment of the discrete neural network error distribution

   $$\left|\alpha_{1g}\right| = 2\left|p_2\Phi_2 - p_1 + p_1\Phi_1\right|$$

   shows that the $|\alpha_{1g}|$ minimization results in satisfying the optimization criterion for the average risk function under the condition that average risk function components are equal for both classes and the following restriction upon coefficients of matrix L:

   $$l_{22} - l_{21} = l_{11} - l_{12}$$

6. The analysis of the expression for the second moment of the discrete neural network error distribution $\alpha_{2g} = 4|P_2\Phi_2 + P_1 - P_1\Phi_1|$ shows that the α_{2g} minimization results in satisfying the optimization criterion for the average risk function under the previous conditions.
7. Additional limitations related to the finite number of considered moments for $|\alpha_{1g}|$ and α_{2g} and $p_1r_1 = p_2r_2$ for $|\alpha_{1g}|$ make these functionals a single-extremum under the limited structure of the open-loop system and multi-modal input signal distribution. Functional α_{2g} can be a multi-extremum in the general case local minimum for the average risk function and $l_{22} - l_{21} = l_{11} - l_{12}$.
8. In the case of an arbitrary open-loop neural network structure (arbitrary divisional surface) according to p. 7.1 for $b_1 = b_2 = 1$ and $c_1 = c_2 = 1$, one obtains

   $$\alpha_{2g} = 4\left[p_2\Phi_2 + p_1 - p_1\Phi_1\right]$$

   where

   $$\Phi_k = \Phi_k\left[S(\mathbf{x})\right] = \overbrace{\int \cdots \int}^{N}_{S(\mathbf{x})<0} f_k(\mathbf{x})\mathrm{d}\mathbf{x}, \quad k = 1, 2$$

Here functional α_{2g} is proportional to the average risk function under the arbitrary neural network structure (two pattern classes, two solutions) and aforementioned limitations upon matrix L.

9. The consideration of the aforementioned functionals of secondary optimization is interesting in spite of the limitations because it results in the sufficiently simple adjustable system implementation with the closed-cycle adjustment, and it can be useful in the design of the neural network with flexible structure.

7.3 About Selection of the Secondary Optimization Functional in the "Adalin" System

The basis of the closed-cycle adjustment methods represented in the works of Widrow [7-1] in the so-called "Adalin" systems is the minimization of the second moment of analogous error distribution. He used the following rule:

> It can be shown using some geometric arguments that the mean square of discrete error is a monotone function of the mean square of analogous error and their minimization is a minimization of the average risk function.

This rule is not correct because the minimization of the average risk function for Gaussian distributions with different covariance matrices is performed with the help of the second order divisional surface. Let us consider one neuron system. Then the coincidence of the optimal solutions by the criteria of α_{2g} and α_{2a} minimization takes place only in the case of one and the same covariance matrices corresponding to the first and second pattern classes [7-2].

Let us analyze the extremum properties of the second order moments for analogous and discrete errors of a one-dimensional neuron in order to estimate the difference between the optimal solutions by the criteria of α_{2g} and α_{2a} minimization.

To do that, one must *(a)* calculate the minimizing coefficients a_0' and a_1' for α_{2a}; *(b)* calculate the minimizing coefficients a_0'' and a_1'' for α_{2g}; *(c)* calculate the difference $\Delta\alpha_{2a} = \alpha_{2a}(a_0', a_1') - \alpha_{2g}(a_0'', a_1'')$.

Figure 7.1 shows the dependence $\Delta R = \Delta\alpha_{2g}(\mu_2)$ for some particular case from [7-2] for two normal distributions with fixed mathematical expectations, one of which has a changing variance μ_2 for the distribution of one of classes. The limitation of α_{2a} minimum criterion is well illustrated by the example of multi-modal distributions

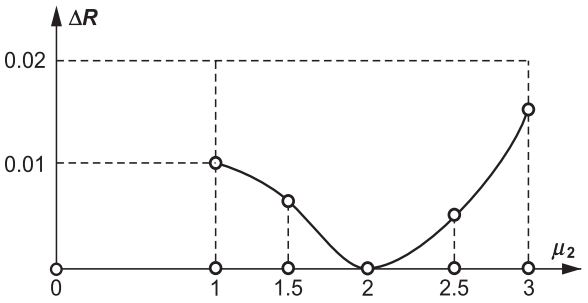


Fig. 7.1. Comparison of the α_{2a} and α_{2g} minimum criteria

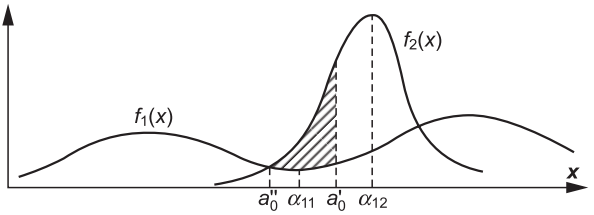


Fig. 7.2.
Comparison of the α_{2a} and α_{2g} minimum criteria for multi-modal distributions

(Fig. 7.2). Here the neuron thresholds a_0' and a_1' optimized by the α_{2a} and α_{2g} minimum criteria are shown. The crosshatched area is the difference ΔR between two criteria.

7.4 Development of the Secondary Optimization Functionals Corresponding to the Given Primary Optimization Criterion

The secondary optimization functional development is performed below for the case of the open-loop neural networks with the arbitrary structure with $K_p = K = 2$, i.e., with the divisional surface of the arbitrary form.

7.4.1 The Average Risk Function Minimum Criterion

The main problem consists in the selection of the neural network discrete error transformation $x_0(n) = \varepsilon(n) - x_k(n)$ to obtain the discrete error $x_0'(n)$ with the second moment of distribution equal to the average risk function. Let us multiply $x_0(n)$ by A if $\varepsilon(n) = -1$, by B if $\varepsilon(n) = 1$, and then add C to the result. Let us determine parameters (A, B, C) in a way that the second moment of distribution $f_{x_g'}(x_g')$ is equal to R:

$$f_{x_g'}\left(x_g'\right) = \begin{cases} p_1\left(1-\Phi_1\right) \;, & \text{if} \quad x_g' = -2A + C \\ p_1\Phi_1 + \left(1-\Phi_2\right)p_2 \;, & \text{if} \quad x_g' = C \\ p_2\Phi_2 \;, & \text{if} \quad x_g' = 2B + C \end{cases} \tag{7.11}$$

$$\alpha_{2g} = p_1(2A-C)^2 + p_2C^2 + p_1\Phi_1\left(4AC - 4A^2\right) + p_2\Phi_2\left(4BC - 4B^2\right) \tag{7.12}$$

$$R = p_1l_{12} + p_2l_{22} + p_1(l_{11} - l_{12})\Phi_1 + p_2\Phi_2(l_{21} - l_{22}) \tag{7.12a}$$

Consequently,

$$\begin{aligned} C &= \sqrt{p_2l_{22} + 2p_1l_{12} - p_1l_{11}} \\ A &= \frac{\sqrt{p_2(l_{11} + l_{22}) + l_{12}(p_1 - p_2)} + \sqrt{p_2l_{22} + 2p_1l_{12} - p_1l_{11}}}{2} \\ B &= \frac{\sqrt{l_{21} - p_1l_{22} + 2p_1l_{12} - p_1l_{11}} - \sqrt{p_2l_{22} + 2p_1l_{12} - p_1l_{11}}}{2} \end{aligned} \tag{7.13}$$

To obtain parameters A, B, C providing the coincidence of α_{2g} and R within constant direct-current component $(p_1 l_{12} + p_2 l_{22})$, one must put

$$C = 0 \;\; ; \quad A = \frac{1}{2}\sqrt{l_{12} - l_{11}} \;\; ; \quad B = \frac{1}{2}\sqrt{l_{21} - l_{22}} \tag{7.14}$$

It is possible to use the following transformation (Z) for the discrete error:

$$x'_g(n) = \begin{cases} z_{12} \;, & \text{if} \quad x_g = -2, \varepsilon = -1 \\ z_{11} \;, & \text{if} \quad x_g = 0, \varepsilon = -1 \\ z_{22} \;, & \text{if} \quad x_g = 0, \varepsilon = 1 \\ z_{21} \;, & \text{if} \quad x_g = 2, \varepsilon = 1 \end{cases}$$

In this case,

$$f_{x'_g}\left(x'_g\right) = \begin{cases} p_1(1-\Phi_1) \;, & \text{if} \quad x'_g = Z_{12} \\ p_1\Phi_1 \;, & \text{if} \quad x'_g = Z_{11} \\ p_2(1-\Phi_2) \;, & \text{if} \quad x'_g = Z_{22} \\ p_2\Phi_2 \;, & \text{if} \quad x'_g = Z_{21} \end{cases} \tag{7.14a}$$

and the conditions for α_{2g} and R coincidence are

$$Z_{k_p k} = \sqrt{l_{k_p k}} \tag{7.15}$$

7.4.2 Minimum criterion for R under the condition $p_1 l_1 = p_2 r_2$

The minimization for R under the condition $p_1 l_1 = p_2 r_2$, i.e., under the condition

$$p_1 l_{11}\Phi_1 + p_1 l_{12}(1-\Phi_1) - p_2 l_{21}\Phi_2 - p_2 l_{22}(1-\Phi_2) = 0 \tag{7.16}$$

is an equivalent of the Lagrangian functional minimization:

$$R^* = \left[p_1 l_{11}\Phi_1 + p_1 l_{12}(1-\Phi_1)\right](1+\lambda) + \left[p_2 l_{21}\Phi_2 + p_2 l_{22}(1-\Phi_2)\right](1-\lambda) \tag{7.17}$$

The conditions for parameters (A, B, C) providing α_{2g} and R coincidence are

$$\begin{aligned} C &= \sqrt{p_2 l_{22}(1-\lambda) + 2p_1 l_{12}(1+\lambda) - p_1 l_{11}(1+\lambda)} \\ A &= \frac{1}{2}\sqrt{(1+\lambda)\left[p_2 l_{11} + (p_1 - p_2) l_{12}\right] + p_2 l_{22}(1-\lambda)} \\ &\quad + \frac{1}{2}\sqrt{p_2 l_{22}(1-\lambda) + 2p_1 l_{12}(1+\lambda) - p_1 l_{11}(1+\lambda)} \\ B &= \frac{1}{2}\sqrt{(1-\lambda)(l_{21} - p_1 l_{22}) + (1-\lambda)(2p_1 l_{12} - p_1 l_{11})} \\ &\quad - \frac{1}{2}\sqrt{p_2 l_{22}(1-\lambda) + 2p_1 l_{12}(1+\lambda) - p_1 l_{11}(1+\lambda)} \end{aligned} \tag{7.18}$$

The left side of (7.16) represents gradient of R^* by λ. This value can be estimated as the first moment of the transformed (A_1, B_1, C_1) discrete error. The transformation parameters A_1, B_1, and C_1 are obtained in the following way. It follows from (7.11)

$$\alpha_{1g} = p_1(C_1 - 2A_1) + C_1 p_2 + p_1 \Phi_1 2A_1 + p_2 \Phi_2 2B_1 \tag{7.19}$$

The comparison between (7.19) and (7.16) gives

$$A_1 = \frac{l_{11} - l_{21}}{2} \;\; ; \quad B_1 = \frac{l_{22} - l_{21}}{2} \;\; ; \quad C_1 = p_1 l_{11} - p_2 l_{22} \tag{7.20}$$

The use of the previously represented Z-transformation and comparison of (7.14a) and (7.17) gives the following condition for their coincidence:

$$Z_{12} = \sqrt{l_{12}(1+\lambda)} \;\; ; \quad Z_{22} = \sqrt{l_{22}(1-\lambda)} \;\; ; \quad Z_{11} = \sqrt{l_{11}(1+\lambda)} \;\; ; \quad Z_{21} = \sqrt{l_{21}(1+\lambda)} \tag{7.21}$$

The transformation for the discrete error with the first moment of distribution determined by the left side of (7.16) has the following parameters:

$$Z'_{12} = l_{12} \;\; ; \quad Z'_{22} = -l_{22} \;\; ; \quad Z'_{21} = -l_{21} \;\; ; \quad Z'_{11} = l_{11} \tag{7.22}$$

7.4.3
The Minimum Criterion for R under the Condition $p_1 r_1 = \alpha = \text{Const.}$

The minimum criterion for R under the condition $p_1 r_1 = \alpha = \text{const.}$, i.e., under the condition

$$p_1 l_{11} \Phi_1 + p_1 l_{12}(1 - \Phi_1) - \alpha = 0 \tag{7.23}$$

is equivalent to Lagrangian functional minimization:

$$R^* = p_1 l_{12}(1+\lambda) + p_2 l_{22} + p_1(l_{11} - l_{12})(1+\lambda)\Phi_1 + p_2(l_{21} - l_{22})\Phi_2 - \lambda\alpha \tag{7.24}$$

Expressions (7.24) and (7.12a) give the following conditions for parameters A, B, and C providing α_{2g} and R coincidence

$$\begin{aligned}
C &= \sqrt{p_2 l_{22} + 2p_1 l_{12}(1+\lambda) - p_1 l_{11}(1+\lambda) - \alpha\lambda} \\
A &= \frac{1}{2}\sqrt{(1+\lambda)\left[p_2 l_{11} + (p_1 - p_2) l_{12}\right] + p_2 l_{22} - \alpha\lambda} \\
&\quad + \frac{1}{2}\sqrt{p_2 l_{22} + 2p_1 l_{12}(1+\lambda) - p_1 l_{11}(1+\lambda) - \alpha\lambda} \\
B &= \frac{1}{2}\sqrt{l_{21} - p_1 l_{22} + (1+\lambda)(2p_1 l_{12} - p_1 l_{11}) - \alpha\lambda} \\
&\quad - \frac{1}{2}\sqrt{p_2 l_{22} + 2p_1 l_{12}(1+\lambda) - p_1 l_{11}(1+\lambda) - \alpha\lambda}
\end{aligned} \tag{7.25}$$

The transformation parameters A_1, B_1, and C_1 for the discrete error providing coincidence of moment (7.12) and the left side of (7.23) are

$$A_1 = \frac{l_{11} - l_{12}}{2} \;\; ; \quad B_1 = 0 \;\; ; \quad C_1 = p_1 l_{11} - \alpha \tag{7.26}$$

The use of the previously represented Z-transformation and comparison of (7.14a) and (7.24) gives the following condition for their coincidence:

$$Z_{21} = \sqrt{l_{21}} \;\; ; \;\; Z_{22} = \sqrt{l_{22}} \;\; ; \;\; Z_{12} = \sqrt{l_{12}(1+\lambda) - \frac{\alpha\lambda}{p_1}} \;\; ; \;\; Z_{11} = \sqrt{l_{11}(1+\lambda) - \frac{\alpha\lambda}{p_1}} \tag{7.27}$$

The determination of R^* gradient by λ is performed by discrete error $x'_g(n)$ formation with the distribution first moment (7.23) and the following parameters of Z-transformation:

$$Z'_{21} = Z'_{22} = 0 \;\; ; \;\; Z'_{12} = l_{12} - \frac{\alpha}{p_1} \;\; ; \;\; Z'_{11} = l_{11} - \frac{\alpha}{p_1} \tag{7.28}$$

7.5 Neural Network Continuum Models

Let us consider a forming procedure for the secondary optimization functional in the case of a neural network continuum model. The minimum average risk function criterion will be used because the extension to other optimization criteria is not difficult.

The problem of the optimization functional will be solved for the neural network with the arbitrary structure and will be illustrated on some concrete example as it was done above.

7.5.1 Neural Network with a Solution Continuum; Two Pattern Classes

The discrete error transformation has the form

$$x'_g(n) = \begin{cases} Z_2\left[x_g(n)\right] \;\; , & \text{if} \quad \varepsilon(n) = 1 \\ Z_1\left[x_g(n)\right] \;\; , & \text{if} \quad \varepsilon(n) = -1 \end{cases}$$

Consequently, the distribution of the transformed error is

$$f_{x'_g}(x'_g) = p_1 f_{1x_g}\left[Z_1^{-1}(x'_g)\right]\left|\frac{\mathrm{d}Z_1^{-1}(x'_g)}{\mathrm{d}x'_g}\right| + p_2 f_{2x_g}\left[Z_2^{-1}(x'_g)\right]\left|\frac{\mathrm{d}Z_2^{-1}(x'_g)}{\mathrm{d}x'_g}\right|$$

and the expression for the second moment of this distribution after transformation of variables under the condition of monotone form of Z_1 and Z_2 is

$$\alpha_{2g} = \int_{-\infty}^{\infty} \left[Z_1(x_g)\right]^2 p_1 f_{1x_g}(x_g)\mathrm{d}x_g + \int_{-\infty}^{\infty} \left[Z_2(x_g)\right]^2 p_2 f_{2x_g}(x_g)\mathrm{d}x_g$$

The relationships $y = P(\mathbf{x})$, $x_g = \varepsilon - P(\mathbf{x})$ are valid for the neural network with the arbitrary structure. Then

$$x_N = \begin{cases} P_1'\left(x_g, P, x_1, ..., x_{N-1}\right) , & \text{if} \quad \varepsilon = -1 \\ P_2'\left(x_g, P, x_1, ..., x_{N-1}\right) , & \text{if} \quad \varepsilon = 1 \end{cases}$$

The discrete error distribution for the k-th class patterns has the form

$$f_{kx_g}(x_g) = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N-1} f_k\left[x_1, \dots, x_{N-1}, P_k'(x_g, P, x_1, \dots, x_{N-1})\right] \times \left|\frac{\mathrm{d}P_k'(x_g, P, x_1, \dots, x_{N-1})}{\mathrm{d}x_g}\right| \mathrm{d}x_{N-1} \dots \mathrm{d}x_1$$

The following expression for the second moment of discrete error distribution can be obtained after the corresponding transformations:

$$\alpha_{2g} = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left\{Z_1[-1 - P(\mathbf{x})]\right\}^2 p_1 f_1(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left\{Z_2[-1 - P(\mathbf{x})]\right\}^2 p_2 f_2(\mathbf{x})\mathrm{d}\mathbf{x} \quad (7.29)$$

It can be shown that in the particular case of the neuron with a continuum of solutions, one gets using (7.8a)

$$\alpha_{2g} = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left\{Z_1\left[-1 - F\left(\sum_{i=1}^{N} a_i x_i - a_0\right)\right]\right\}^2 p_1 f_1(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left\{Z_2\left[-1 - F\left(\sum_{i=1}^{N} a_i x_i - a_0\right)\right]\right\}^2 p_2 f_2(\mathbf{x})\mathrm{d}\mathbf{x}$$

In the general case,

$$\alpha_{2g} = \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left\{Z_1\left[x_g(\mathbf{x})\right]\right\}^2 p_1 f_1(\mathbf{x})\mathrm{d}\mathbf{x} + \overbrace{\int_{-\infty}^{\infty} \dots \int}^{N} \left\{Z_2\left[x_g(\mathbf{x})\right]\right\}^2 p_2 f_2(\mathbf{x})\mathrm{d}\mathbf{x}$$

The comparison of this expression with the expression for the average risk function

$$R=\overbrace{\int_{-\infty}^{\infty}\dots\int}^{N} p_1 f_1(\mathbf{x}) l_1\left[x_k=P(\mathbf{x})\right]+p_2 f_2(\mathbf{x}) l_2\left[x_k=P(\mathbf{x})\right]\mathrm{d}\mathbf{x}$$

gives the relationships for the discrete error required for α_{2g} and R equality:

$$Z_1\left(x_g\right)=\sqrt{l_1\left(-1-x_g\right)}\quad;\quad Z_2\left(x_g\right)=\sqrt{l_2\left(1-x_g\right)} \tag{7.30}$$

7.5.2
Neural Network with a Solution Continuum; Continuum of Pattern Classes

In this case,

$$f_{x_g'}\left(x_g'\right)=f_{x_g}\left[Z^{-1}\left(x_g'\right)\right]\frac{\mathrm{d}Z^{-1}\left(x_g'\right)}{\mathrm{d}x_g'}$$

Consequently, under the condition of monotone form of the function $Z(x_g)$

$$\alpha_{2g}=\int_{-\infty}^{\infty}\left[Z\left(x_g\right)\right]^2 f_{x_g}\left(x_g\right)\mathrm{d}x_g \tag{7.31}$$

Here

$$y=P(\mathbf{x})\quad;\quad x_g=\varepsilon-P(\mathbf{x})\quad;\quad x_N=P'(x_g,P,\varepsilon,x_1,\dots,x_{N-1})$$

$$f_{x_g}=\overbrace{\int_{-\infty}^{\infty}\dots\int}^{N} f\left[x_1,\dots,x_{N-1},P'(x_g,P,\varepsilon,x_1,\dots,x_{N-1})\middle|\varepsilon\right]$$
$$\times f_\varepsilon(\varepsilon)\left|\frac{\mathrm{d}P'(x_g,P,\varepsilon,x_1,\dots,x_{N-1})}{\mathrm{d}x_g}\right|\mathrm{d}x_{N-1}\,\mathrm{d}x_1\,\mathrm{d}\varepsilon$$

Then one gets after the corresponding transformation of variables and using (7.31)

$$\alpha_{2g}=\int_{-\infty}^{\infty}\overbrace{\int_{-\infty}^{\infty}\dots\int}^{N}\left\{Z\left[\varepsilon-P(\mathbf{x})\right]\right\}^2 f(\mathbf{x}|\varepsilon)f_\varepsilon(\varepsilon)\mathrm{d}\varepsilon\mathrm{d}\mathbf{x} \tag{7.32}$$

It can be shown that in the particular case of the neuron with a continuum of solutions, one gets using (7.7a)

$$\alpha_{2g}=\int_{-\infty}^{\infty}\overbrace{\int_{-\infty}^{\infty}\dots\int}^{N}\left\{Z\left[\varepsilon-F\left(\sum_{i=1}^{N}a_i x_i-a_0\right)\right]\right\}^2 f(\mathbf{x}|\varepsilon)f_\varepsilon(\varepsilon)\mathrm{d}\varepsilon\mathrm{d}\mathbf{x}$$

The comparison of this expression with the expression for the average risk function

$$R=\int_{-\infty}^{\infty}\overbrace{\int_{-\infty}^{\infty}\cdots\int}^{N} f_{\varepsilon}(\varepsilon)f(\mathbf{x}|\varepsilon)l\left[x_k=P(\mathbf{x}),\varepsilon\right]\mathrm{d}\mathbf{x}\,\mathrm{d}\varepsilon$$

provides the relation for the discrete error required for α_{2g} and R equality:

$$Z\left(x_g\right)=\sqrt{l\left[\left(\varepsilon-x_g\right),\varepsilon\right]}$$

7.5.3 Neural Network with K_p Solutions; K Pattern Classes

The Eq. (7.10a) gives the expression for the distribution of the transformed discrete error for the neuron with K_p solutions in the case of the k-th pattern class at $x'_g=(k-k_p)A_{k_pk}$:

$$f_{kx'_g}(x'_g)=\Phi_k\left(\frac{a_{k_p,k_p+1}+a_0}{a_N};\frac{a_1}{a_N};\ldots;\frac{a_{N-1}}{a_N}\right)-\Phi_k\left(\frac{a_{k_p-1,k_p}+a_0}{a_N};\frac{a_1}{a_N};\ldots;\frac{a_{N-1}}{a_N}\right)$$

Consequently,

$$\alpha_{2g}=\sum_{k=1}^{K}\sum_{k_p=1}^{K}\left[(k-k_p)A_{k_pk}\right]^2 p_k \times\left[\Phi_k\left(\frac{a_{k_p,k_p+1}+a_0}{a_N};\frac{a_1}{a_N},\ldots,\frac{a_{N-1}}{a_N}\right)-\Phi_k\left(\frac{a_{k_p-1,k_p}+a_0}{a_N};\frac{a_1}{a_N}\right)\right]$$

In the case of the neural network with the arbitrary structure,

$$\alpha_{2g}=\sum_{k=1}^{K}\sum_{k_p=1}^{K_p}\left[(k-k_p)A_{k_pk}\right]^2 p_k\int_{S^{(k_p)}(\mathbf{x})>0} f_k(\mathbf{x})\mathrm{d}\mathbf{x}$$

The comparison of this expression with the expression for the average risk function

$$R=\sum_{k=1}^{K}\sum_{k_p=1}^{K_p} l_{k_pk}P_k\int_{S^{(k_p)}(\mathbf{x})>0} f_k(\mathbf{x})\mathrm{d}\mathbf{x}$$

provides the relation for the discrete error required for α_{2g} and R equality:

$$A_{k_pk}=\frac{1}{k-k_p}\sqrt{l_{k_pk}}$$

7.5.4
Neural Network with N^* Output Channels; K_0 Gradations in Each Class

The distribution function for the discrete error for the assemblage of patterns of $(k_1,\ldots, k_{N^*})$-class is

$$f'_{(k_1,\ldots,k_{N^*})}(x_{1g},\ldots,x_{N^*g}) = \overbrace{\int\limits_{S^{(k_{1\mathrm{p}},\ldots k_{N^*\mathrm{p}})}(\mathbf{x})>0}\cdots\int}^{N} f'_{(k_1,\ldots,k_{N^*})}(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

at $(x_{1g},\ldots, x_{N^*g}) = (k_1,\ldots, k_{N^*}) - (k_{1\mathrm{p}},\ldots, k_{N^*\mathrm{p}})$.

Let us use the following transformation for the vector $(x_{1g},\ldots, k_{N^*})$ required for the expression of the transformed discrete error $x'_g(n)$. Let us multiply vector $\mathbf{x}_g$ by the scalar $A(k_1, \ldots, k_{N^*}, k_{1\mathrm{p}}, \ldots, k_{N^*\mathrm{p}})$ and calculate the sum of squares of the resultant vector components. The result will be the transformed discrete error $x'_g(n)$. For the pattern class assemblage,

$$M\left[x'^2_{1g}+\ldots+x'^2_{N^*g}\right] = \sum_{k_1=1}^{K_0}\cdots\sum_{k_{N^*}=1}^{K_0}\sum_{k_{1\mathrm{p}}=1}^{K_0}\cdots\sum_{K_{N^*\mathrm{p}}=1}^{K_0}\left[(k_1,\ldots,k_{1\mathrm{p}})^2+(k_{N^*}-k_{N^*\mathrm{p}})^2\right]$$

$$\times A^2(k_1,\ldots,k_{N^*},k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}})f_{\varepsilon}(k_1,\ldots,k_{N^*})\overbrace{\int\limits_{S^{(k_{1\mathrm{p}},\ldots k_{N^*\mathrm{p}})}(\mathbf{x})>0}\cdots\int}^{N} f_{(k_1,\ldots,k_{N^*})}(\mathbf{x})\,\mathrm{d}\mathbf{x}$$

The comparison of this expression with the expression for the average risk function provides the relation for the transformation A parameters in the following form:

$$A\left(k_1,\ldots,k_{N^*},k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}}\right) = \sqrt{\frac{l\left(k_1,\ldots,k_{N^*},k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}}\right)}{\left(k_1-k_{1\mathrm{p}}\right)^2+\ldots+\left(k_{N^*},\ldots,k_{N^*\mathrm{p}}\right)^2}} \tag{7.33}$$

This transformation makes equal the values $M[x'^2_{1g}+\ldots+x'^2_{N^*g}]$ and R.

7.5.5
Neural Network with N^* Output Channels – Neural Network Solution Continuum

In this case, vector $\mathbf{x}_g = \varepsilon - \mathbf{P}(\mathbf{x})$ is the vector of dimensionality N^*. The distribution density of the sum of squares of the vector $\mathbf{x}_g$ components has the form

$$f_{x_g^2}(x_g^2) = \overbrace{\int\limits_E\cdots\int}^{N^*}\overbrace{\int\limits_{X'}\cdots\int}^{N-1} f\left[x_1,\ldots,x_{N-1},P'(*)\middle|\varepsilon\right]f_{\varepsilon}(\varepsilon)\left|\frac{\mathrm{d}P'(*)}{\mathrm{d}x_g^2}\right|\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1\,\mathrm{d}\varepsilon$$

$$x_N = P'\left(x_g^2,P_1,\ldots,P_{N^*},\varepsilon_1,\ldots,\varepsilon_{N^*},x_1,\ldots,x_N\right) = P'(*)$$

The distribution of the square of the transformed discrete error is

$$f_{x'_g}\left(x'^2_g\right)=f_{x^2_g}\left[Z^{-1}\left(x'^2_g\right)\right]\frac{\mathrm{d}Z^{-1}\left(x'^2_g\right)}{\mathrm{d}x'^2_g}$$

Consequently, the first moment of this distribution is

$$\alpha_{1g}=\int\limits_{-\infty}^{\infty}\left[Z(x^2_g)\right]f_{x^2_g}(x^2_g)\mathrm{d}x^2_g=\int\limits_{-\infty}^{\infty}\left[Z(x^2_g)\right]\overbrace{\int\limits_{E}\ldots\int}^{N^*}\overbrace{\int\limits_{X'}\ldots\int}^{N-1}f\left[x_1,\ldots,x_{N-1},P'(*)\middle|\varepsilon\right]$$
$$f_{\varepsilon}(\varepsilon)\left|\frac{\mathrm{d}P'(*)}{\mathrm{d}x^2_g}\right|\mathrm{d}x_{N-1}\ldots\mathrm{d}x_1\,\mathrm{d}\varepsilon\mathrm{d}x^2_g$$

After the transformation of variables,

$$\alpha_{1g}=\overbrace{\int\limits_{E}\ldots\int}^{N^*}\overbrace{\int\limits_{X}\ldots\int}^{N}\left\{\sum_{i^*=1}^{N^*}\left[\varepsilon_{i^*}-P_{i^*}(\mathbf{x})\right]^2\right\}f(\mathbf{x}|\varepsilon)f_{\varepsilon}(\varepsilon)\mathrm{d}\varepsilon\mathrm{d}\mathbf{x}$$

The comparison of this expression with the expression for the average risk function in the case of the neural networks with N^* output channels and a continuum of solutions gives the following form for the equation of discrete error transformation function:

$$Z\left\{\sum_{i^*=1}^{N^*}\left[\varepsilon_{i^*}-P_{i^*}(\mathbf{x})\right]^2\right\}=l\left[\mathbf{P}(\mathbf{x}),\varepsilon\right] \tag{7.34}$$

7.6 Neural Network in the Self-Learning Mode and Arbitrary Teacher Qualification

The expression for the average risk function in the self-learning mode in the case of K_p solutions has the form

$$R=\sum_{k_\mathrm{p}=1}^{K_\mathrm{p}}\int\limits_{S\left(k_\mathrm{p}\right)_{(\mathbf{x})>0}}\rho\left[\mathbf{x}-\mathbf{b}_{k_\mathrm{p}}\right]f(\mathbf{x})\mathrm{d}\mathbf{x}$$

It can be shown that in the case of the system with K_p solutions, the transformation of the output signal y forming the signal y' with the first moment of its distribution equal to R is

$$y'=\rho\left[\mathbf{x}-\mathbf{b}(y)\right] \tag{7.35}$$

and in the case of the arbitrary teacher qualification,

$$y' = l(y,\varepsilon)\mathbf{b} + \left(1 - \mathbf{b}^2\right)\rho\left[\mathbf{x} - \mathbf{b}(y)\right] \tag{7.36}$$

The Eqs. (7.35) and (7.36) are also valid in the case of the neural network with a continuum of solutions.

Literature

[7-1] Widrow B (1965) Pattern recognition and adaptive control. Foreign radioelectronics 9:87–111
[7-2] Galushkin AI, Zak LS, Tikhov BP (n.y.) To the comparison of optimization criteria for the adaptive pattern recognition systems. Kiev, Kibernetika 6:122–130
[7-3] Galushkin AI (1971) Implementation of the primary optimization criteria in the pattern recognition systems adjustable by the closed-cycle in the learning mode. MIEM proc., 23:191–203
[7-4] Galushkin AI (1974) Multilayer pattern recognition system synthesis. Moscow, Energiya

Chapter 8

Development of Multivariable Function Extremum Search Algorithms

8.1 Procedure of the Secondary Optimization Functional Extremum Search in Multilayer Neural Networks

The secondary optimization functional extremum search is performed in the present study with the help of iteration methods using a gradient search procedure for the local extremum. The problems of stability and convergence of the gradient procedures and the possibility to accelerate the extremum search are considered. The constraints of the equation and inequality types are considered in the case of multilayer neural network implementation.

The multivariable function extremum search methods develop mainly in two lines. The first one includes the design of standard search programs. The function properties in this case are taken in the sufficiently detailed form. Method convergence and precision in the stationary state is usually analyzed without investigation of transient processes dynamics.

The second line includes the design of adaptive system adjustment algorithms. The function is given in the most general form due to the problem specificity and functioning in conditions of poor a priori information about the input signal properties [8-1 to 8-29].

a multilayer neural network is a particular case of the adaptive system. The peculiar properties of the adaptive system design relate to the fact that even in the case of fixed structure of the open-loop neural network, it is impossible to know anything about the form of the secondary optimization functional. One can only know that it has some local extrema to be found in the process of closed-cycle adjustment. The adjustment circuit optimization problem for the multilayer neural network cannot generally be solved in the stage of optimization functional extremum search.

That is why the main content of Chap. 12 is the adjustment circuit optimization in the investigation of closed-loop systems with quality estimation using the current value of the primary optimization functional.

8.2 Analysis of the Iteration Method for the Multivariable Function Extremum Search

a general expression for the calculation of the system state vector in the case of function $Y(\mathbf{a})$ extremum search at the time $n+1$ by the state vector at the time n has the form (for the unit search system memory)

$$\mathbf{a}(n+1)=\mathbf{a}(n)+K^{*}\left.\frac{\partial Y(\mathbf{a})}{\partial \mathbf{a}}\right|_{\mathbf{a}=\mathbf{a}(n)} \tag{8.1}$$

Here $Y(\mathbf{a})$ is the secondary optimization functional; $\mathbf{a}(n)$ is the system state vector (the current argument value for the extremum function); K^* is the $[N^0 \times N^0]$ matrix of coefficients; and N^0 is the dimensionality of vector $\mathbf{a}$.

Selection of the coefficient of matrix K^* determines the rate of convergence and quality of the iteration method.

Procedure (8.1) describes the known search methods: scan-out, steepest descent, gradient, Gausse-Zeydel, Rosenbrock, Pawell, Sawswell, etc.

The main problem is the selection of constraints upon the matrix K^* parameters providing the necessary quality for the function extremum search system. Let us consider a particular form of the neural network quality function:

$$Y(\mathbf{a}) = \mathbf{a}^T A \cdot \mathbf{a} + \mathbf{B}^T \mathbf{a} + C \tag{8.1a}$$

Here a is the matrix of coefficients of the functional $Y(\mathbf{a})$; $\mathbf{B}$ is the vector of coefficients; and C is the coefficient.

Consequently,

$$\frac{\mathrm{d}Y(\mathbf{a})}{\mathrm{d}\mathbf{a}} = 2A\mathbf{a} + \mathbf{B} \quad ; \quad \frac{\partial^2 Y(\mathbf{a})}{\partial \mathbf{a}_i \partial \mathbf{a}_j} = 2A \quad ; \quad i, j = 1, \ldots, N^0 \tag{8.2}$$

Expressions (8.1) and (8.2) give a recurrent expression for the system state vector at step $n + 1$ through the state vector at step n in the following form:

$$\mathbf{a}(n+1) = \mathbf{a}(n) + K^*[2A\mathbf{x}(n) + \mathbf{B}]$$

or

$$\mathbf{a}(n+1) = K^* \cdot \mathbf{B} + \left[Y + 2K^* A\right]\mathbf{a}(n) \tag{8.3}$$

Here Y is the unit matrix.

Let us determine the coefficients of matrix K^* providing the iteration procedure convergence by one step starting from any initial state. The $\mathbf{a}(1)$ value providing the extremum is determined in the following way:

$$\mathbf{a}(1) = -\frac{1}{2} A^{-1} \mathbf{B}$$

Putting it into (8.3), one gets the following optimal matrix K^*:

$$K^*_{\mathrm{opt}} = -\frac{1}{2} A^{-1}$$

The system that provides transition to $(n + 1)$-th step by the results obtained at the n-th step is called stable if the function value at the $(n + 1)$-th step is less than that at the n-th step. In the opposite case this system is called self-oscillating or an unstable one:

$$1)\quad \mathbf{a}(n+1)=K^{*}B+\left[Y+2K^{*}A\right]\mathbf{a}(n)$$

$$2)\quad \mathbf{a}^{T}(n)A\mathbf{a}(n)+B^{T}\mathbf{a}(n)=\left\{\begin{matrix}>\\<\end{matrix}\right\}\mathbf{a}^{T}(n+1)\cdot A\cdot\mathbf{a}(n+1)+B^{T}a(n+1) \tag{8.4}$$

The solution of this system of equations requires the use of a computer. The search system with particular forms of matrix K^{*} (particular methods of extremum search) is stable if matrix K^{*} satisfies (8.4).

Let us obtain the non-recurrent expression for $\mathbf{a}(n)$. It follows from (8.3):

$$\mathbf{a}(1) = K^{*}\mathbf{B} + [Y + 2AK^{*}]\mathbf{a}(0);$$

$$\mathbf{a}(2) = K^{*}\mathbf{B} + [Y + 2AK^{*}]K^{*}\mathbf{B} + [Y + 2AK^{*}]^{2}\mathbf{a}(0);$$

$$\mathbf{a}(3) = K^{*}\mathbf{B} + [Y + 2AK^{*}]K^{*}\mathbf{B} + [Y + 2AK^{*}]^{2}K^{*}\mathbf{B} + [Y + 2K^{*}a]^{3}\mathbf{a}(0);$$

By induction

$$\mathbf{a}(n)=\left[Y+\left(Y+2K^{*}A\right)+\ldots+\left(Y+2K^{*}A\right)^{n-1}\right]K^{*}\mathbf{B}+\left[Y+2K^{*}A\right]^{n}\mathbf{a}(0)$$

Taking into account that

$$Y+\left(Y+2K^{*}A\right)+\ldots+\left(Y+2K^{*}A\right)^{n-1}=\frac{Y-\left[Y+2K^{*}A\right]^{n}}{Y-\left[Y+2K^{*}A\right]}$$

$$=-\left[Y-\left(Y+2K^{*}A\right)^{n}\right]\left(2K^{*}A\right)^{-1}$$

one can write the expression for $\mathbf{a}(n)$ in the following form:

$$\mathbf{a}(n)=\left(Y+2K^{*}A\right)^{n}\mathbf{a}(0)+\frac{1}{2}\left[\left(Y+2K^{*}A\right)^{n}-Y\right](K^{*}A)^{-1}K^{*}\mathbf{B}$$

Then the final non-recurrent expression for the search system state vector is

$$\mathbf{a}(n)=\left(Y+2K^{*}A\right)^{n}\mathbf{a}(0)+\frac{1}{2}\left[\left(Y+2K^{*}A\right)^{n}-Y\right]A^{-1}\mathbf{B} \tag{8.5}$$

Putting the condition of optimum for the operation speed of the search system $K = -0.5A^{-1}$, one obtains the expected result:

$$\mathbf{a}(n)=-\frac{1}{2}A^{-1}\mathbf{B}\quad,\quad n=1,2,\ldots,$$

that corresponds to the extremum function value.

The analysis of the non-recurrent expression gives the constraints upon the matrix K^{*} parameters providing the search iteration procedure convergence.

It follows from (8.5) that

$$\lim_{n\to\infty} \mathbf{a}(n) = -\frac{1}{2}A^{-1}\mathbf{B}$$

i.e., it does not depend on $\mathbf{a}(0)$ and equals the state vector extremum value if

$$\lim_{n\to\infty} \left(Y + 2K^{*}A\right)^{n} = \mathbf{O}$$

where $\mathbf{O}$ is the zero matrix (Y, a). This expression can be used for the corroboration of the search system convergence. The expression for K^{*} matrix providing the self-oscillation mode of the search procedure is given in [8-12].

8.3 About the Stochastic Approximation Method

The stochastic approximation method is realized by the search system that is similar to the gradient one. However, the system parameters (matrix K^{*}) in this case are not fixed [8-1, 8-3, 8-6, 8-8]. The stochastic approximation method is used in the case of random erroneous measurements of minimized function gradient vector. The presence of random errors makes it necessary to introduce variability for the search system parameters in order to provide zero random error for the determination of the extreme point. The disadvantages of this method consisting in the increase of systematic errors in the transient process of the extreme point search are mentioned in [8-29].

The present study deals with the neural network synthesis technique where the stochastic approximation method can be combined with some other search methods with fixed parameters. The closed-loop neural network design is performed under the condition of some initial indeterminateness of matrix K^{*} that is eliminated only in the stage of analysis of the closed-loop system. The question about an optimal selection of matrix K^{*} parameters will be ill-posed in this case because the form of the minimized function is not known in advance.

8.4 Iteration Methods for Multivariable Function Extremum Search in the Case of Equality-Type Constraints upon Variables

The equality-type constraints upon the adjustable neural network coefficients can be written in the general in the form

$$q_{\mu}(\mathbf{a}) = 0 \quad , \quad \mu = 1,\ldots,M_1 \quad , \quad M_1 < N^{0} + 1$$

In the real neural networks,

$$\sum_{i=0}^{N} a_i = a = \text{const.} \tag{8.6}$$

i.e., there are, for example, constraints upon the sum of the coefficients.

8.4.1 Search Algorithm

The problem of quality function $Y(\mathbf{a})$ minimization in this case is solved in the multilayer neural networks by the construction of the Lagrange function

$$Y(\mathbf{a}, \boldsymbol{\lambda}) = Y(\mathbf{a}) + \boldsymbol{\lambda}^T \mathbf{q}^T(\mathbf{a})$$

where $\boldsymbol{\lambda}^T = [\lambda_1, \ldots, \lambda_{M_1}]$ is the vector of Langrangian multipliers; and $\mathbf{q}^T(\mathbf{a}) = [\mathbf{q}_1(\mathbf{a}), \ldots, \mathbf{q}_{M_1}(\mathbf{a})]$ is the vector function of constraints.

The solution of the minimization problem is reduced to the solution of the following system of equations:

$$\frac{\mathrm{d}Y(\mathbf{a}, \lambda)}{\mathrm{d}\mathbf{a}} = \frac{\mathrm{d}Y(\mathbf{a})}{\mathrm{d}\mathbf{a}} + \mathbf{Q}(\mathbf{a})\lambda = 0 \quad ; \quad \frac{\mathrm{d}Y(\mathbf{a}, \lambda)}{\mathrm{d}\lambda} = \mathbf{q}(\mathbf{a}) = 0 \tag{8.7}$$

Here

$$\mathbf{Q}(\mathbf{a}) = \left[\frac{\partial \mathbf{q}_\mu(\mathbf{a})}{\partial a_i}\right] = \begin{bmatrix} \dfrac{\partial q_1(\mathbf{a})}{\partial a_i} & \cdots & \dfrac{\partial q_{M_1}(\mathbf{a})}{\partial a_0} \\ \dfrac{\partial q_1(\mathbf{a})}{\partial a_N} & \cdots & \dfrac{\partial q_{M_1}(\mathbf{a})}{\partial a_N} \end{bmatrix}$$

The recurrent relationship for the search algorithm follows from (8.7):

$$\mathbf{a}(n+1) = \mathbf{a}(n) + K^*_{aa}(n) \frac{\partial Y(\mathbf{a}, \boldsymbol{\lambda})}{\partial \mathbf{a}}\bigg|_{\substack{\mathbf{a}=\mathbf{a}(n) \\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}} + K^*_{a\lambda}(n) \frac{\partial Y(\mathbf{a}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}\bigg|_{\substack{\mathbf{a}=\mathbf{a}(n) \\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}}$$

$$\boldsymbol{\lambda}(n+1) = \boldsymbol{\lambda}(n) + K^*_{\lambda a}(n) \frac{\partial Y(\mathbf{a}, \boldsymbol{\lambda})}{\partial \mathbf{a}}\bigg|_{\substack{\mathbf{a}=\mathbf{a}(n) \\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}} + K^*_{\lambda\lambda}(n) \frac{\partial Y(\mathbf{a}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}\bigg|_{\substack{\mathbf{a}=\mathbf{a}(n) \\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}}$$

The search system in this case can be represented by the equivalent discrete system with parameter matrices $K^*_{aa}, K^*_{a\lambda}, K^*_{\lambda a}, K^*_{\lambda\lambda}$. Taking into account (8.7), the final expression for the search algorithm can be written in the form

$$\mathbf{a}(n+1) = \mathbf{a}(n) + K^*_{aa}(n) \left[\frac{\partial Y(\mathbf{a})}{\partial \mathbf{a}} + Q(\mathbf{a}) \cdot \boldsymbol{\lambda}\right]_{\substack{\mathbf{a}=\mathbf{a}(n) \\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}} + K^*_{a\lambda}(n) \mathbf{q}(\mathbf{a})\big|_{\mathbf{a}=\mathbf{a}(n)}$$

$$\boldsymbol{\lambda}(n+1) = \boldsymbol{\lambda}(n) + K^*_{\lambda a}(n) \left[\frac{\partial Y(\mathbf{a})}{\partial \mathbf{a}} + Q(\mathbf{a}) \cdot \boldsymbol{\lambda}\right]_{\substack{\mathbf{a}=\mathbf{a}(n) \\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}} + K^*_{\lambda\lambda}(n) \mathbf{q}(\mathbf{a})\big|_{\mathbf{a}=\mathbf{a}(n)}$$

In this case of constraints (8.6),

$$\mathbf{a}(n+1)=\mathbf{a}(n)+K^{*}_{aa}(n)\left[\frac{\mathrm{d}Y(\mathbf{a})}{\mathrm{d}\mathbf{a}}\Big|_{\mathbf{a}=\mathbf{a}(n)}+\mathbf{1}\cdot\lambda_1(n)\right]+K^{*}_{a\lambda}(n)\left[\sum_{i=0}^{N^0}a_i(n)-\alpha\right]$$

$$\lambda_1(n+1)=\lambda_1(n)+K^{*}_{\lambda a}(n)\left[\frac{\mathrm{d}Y(\mathbf{a})}{\mathrm{d}\mathbf{a}}\Big|_{\mathbf{a}=\mathbf{a}(n)}+\mathbf{1}\cdot\lambda_1(n)\right]+K^{*}_{\lambda\lambda}(n)\left[\sum_{i=0}^{N^0}a_i(n)-\alpha\right]$$

where $\mathbf{1}$ is the column vector of dimensionality N_0+1 consisting of 1.

8.4.2
Analysis of the Matrix of the Second Derivatives of the Lagrange Function

If $Y(\mathbf{a})$ is determined by (8.1a) and the following designation is introduced: $Y^T=[a_0,\ldots,a_{N^0},\lambda_1,\ldots,\lambda_{M_1}]$, then

$$\left[\frac{\partial^2 Y(\mathbf{a},\lambda)}{\partial y_i\,\partial y_j}\right]=\left[\begin{array}{c|c}\mathrm{I} & \mathrm{III}\\ \hline \mathrm{II} & \mathrm{IV}\end{array}\right]\quad,\quad \mathrm{I}\rightarrow i=0,\ldots,N^0\quad;\quad j=0,\ldots,N^0$$

$$\mathrm{II}\rightarrow i=N^0+1,\ldots,N^0+M_1\quad;\quad j=0,\ldots,N^0$$

$$\mathrm{III}\rightarrow i=0,\ldots,N^0\quad;\quad j=N^0+1,\ldots,N^0+M_1$$

$$\mathrm{IV}\rightarrow i=N^0+1,\ldots,N^0+M_1;\ j=N^0+M_1$$

$$i,j=0,\ldots N^0\quad,\quad N+1,\ldots,N^0+M_1$$

It is clear that [I] = 2A, [III] = [II]^T = $\mathbf{Q}(\mathbf{a})$, [IV] = 0.

Consequently, the matrix of the second derivatives of the Lagrange function has the following form:

$$\left[\frac{\partial^2 Y(\mathbf{a},\lambda)}{\partial y_i\,\partial y_j}\right]=\left[\begin{array}{c|c}2A & Q\\ \hline Q^T & 0\end{array}\right]\quad,\quad i,j=0,\ldots,N^0,N^0+1,\ldots,N^0+M_1$$

8.4.3
Operation Speed Optimization for the Extremum Search Iteration Procedure in the Case of Equality-Type Constraints

Using the Newton method for minimization of the Lagrange function, one obtains the following conditions for the optimal operation speed:

$$K^{*}(n)=\left[\begin{array}{c|c}K^{*}_{aa}(n) & K^{*}_{a\lambda}(n)\\ \hline K^{*}_{\lambda a}(n) & K^{*}_{\lambda\lambda}(n)\end{array}\right]=-\left[\frac{\partial^2 Y(\mathbf{y})}{\partial y_i\,\partial y_j}\right]^{-1}_{\mathbf{y}=\mathbf{y}(n)}\quad,\quad i,j=0,\ldots,N^0,N^0+1,\ldots,N^0+M_1$$

It can be shown that the existence condition for the matrix, inverse to the matrix of the input derivatives of the Lagrange function, is the condition of equality of M_1 to the rank of matrix Q. Consequently,

$$\begin{bmatrix} 2A & Q \\ Q^T & 0 \end{bmatrix}^{-1} = \left[\begin{array}{c|c} A_1^{-1}+A_1^{-1}QH^{-1}Q^TA_1^{-1} & -A_1^{-1}QH^{-1} \\ \hline -H^{-1}Q^TA_1^{-1} & H^{-1} \end{array}\right], \quad i,j=0,\ldots,N^0,N^0+1,\ldots,N^0+M_1$$

where $H = -Q^T{A_1}^{-1}Q$, $A_1 = 2A$. The expressions for matrices $K^*_{aa}(n)$, $K^*_{a\lambda}(n)$, $K^*_{\lambda a}(n)$, and $K^*_{\lambda\lambda}(n)$, providing the optimal operation speed for the search procedure, can be derived from the above equations.

8.4.4
Optimal Operation Speed under Constraints (8.6)

In this case $K^*_{aa}(n) = -A_1^{-1}[I + L]$, where I is a unit matrix of dimensionality, $[(N^0+1)\times(N^0+1)]$; $L = QH^{-1}Q^TA_1^{-1} = Q(-Q^TA_1^{-1}Q)^{-1}Q^TA_1^{-1}$.

In the particular case of $Q^T = [1,\ldots,1]$,

$$H^{-1} = -\frac{1}{\sigma_A}, \quad \text{where} \quad \sigma_A = \sum_{i=1}^{N^0}\sum_{j=1}^{N^0} a'_{ij} \; ; \quad A_1^{-1} = \left[a'_{ij}\right]$$

$$L = -\frac{1}{\sigma_A}QQ^TA_1^{-1} = -\frac{1}{\sigma_A}\begin{bmatrix} \alpha_0 & \ldots & \alpha_{N^0} \\ \alpha_0 & \ldots & \alpha_{N^0} \end{bmatrix} \; ; \quad \alpha_j = \sum_{i=0}^{N^0} a_{ij}, \quad j=0,\ldots,N^0$$

It must be mentioned that at any a priori information, matrix K^*_{aa} differs from matrix $(-A_1^{-1})$, and it is non-diagonal even if matrix a is diagonal. Matrix $K^*_{\lambda\lambda}$ has the following form:

$$K^*_{\lambda\lambda} = H^{-1} = -\frac{1}{\sigma_A}$$

i.e., under one constraint, the optimal $K^*_{\lambda\lambda}$ is determined only by the sum of elements of matrix A_1^{-1} across matrix rows and columns. In this case,

$$K^*_{\alpha\lambda} = A_1^{-1}QH^{-1} = A_1^{-1}Q\left(-Q^TA_1^{-1}Q\right)^{-1} = A_1^{-1}Q\left(-\frac{1}{\sigma_A}\right) = -\frac{1}{\sigma_A}\begin{bmatrix} \alpha_0 \\ \alpha_{N^0} \end{bmatrix}$$

Matrix $K^*_{a\lambda}$ is not a zero matrix under any a priori information about matrix a, i.e., the cross-connections are present in the search algorithm.

8.4.5
The Case of Constraints of Equality Type That Can Be Solved

When constraints have the linear form

$$Q^T\mathbf{a} = \alpha \;, \quad \text{where} \quad Q^T = \begin{bmatrix} Q_{01} & \ldots & Q_{N^0 1} \\ \ldots & \ldots & \ldots \\ Q_{0M_1} & \ldots & Q_{N^0 M_1} \end{bmatrix} \tag{8.8}$$

then one can solve the system of equations for M_1 variables and express coefficients $a_0, \ldots, a_{M_1-1}$ through remained $(N_0 + 1 - M_1)$ variables. Matrix Q in this case is divided into two blocks:

$$Q^T = \left[Q_1^T, Q_2^T\right] = \left[\begin{array}{c|c} Q_{01}\cdots Q_{M_1 1} & Q_{(M_1+1)1}\cdots Q_{N^0 1} \\ \hline Q_{0M_1}\cdots Q_{M_1 M_1} & Q_{(M_1+1)M1}\cdots Q_{N^0 M_1} \end{array}\right]$$

Then constraints (8.8) take the following form:

$$\mathbf{Q}_1^T \mathbf{a}^{(1)} + \mathbf{Q}_2^T \mathbf{a}^{(2)} = \alpha$$

where

$$\mathbf{a}^{(1)T} = \left[a_0, \ldots, a_{M_1-1}\right] \quad , \quad \mathbf{a}^{(2)T} = \left[a_{M_1}, \ldots, a_{M^0}\right]$$

$$\mathbf{a}^{(1)} = \left(Q^T\right)^{-1}\left(\alpha - Q_2^T \mathbf{a}^{(2)}\right) \tag{8.9}$$

This expression is substituted into $Y(\mathbf{a})$ and the extremum of the resultant function of $(N_0 + 1 - M_1)$ variables is found by the previously described method. Optimal values of $(N_0 + 1 - M_1)$ variables are determined. Then optimal values of M_1 variables are determined according to (8.9). Taking into account (8.6), the expression (8.9) takes the form

$$a_0 = \alpha - \sum_{i=1}^{N^0} a_i$$

8.4.6
Iteration Process Stability under Equality-Type Constraints

We shall consider the search process to be stable as the Lagrange function value decreases at each step, i.e.,

$$Y[\mathbf{y}(n)] < Y[\mathbf{y}(n-1)] \tag{8.10}$$

Transforming $Y(y)$ into the Taylor series in the neighborhood of point $y(n–1)$ and throwing away the terms of higher than the second order, one obtains

$$Y\left[\mathbf{y}(n-1) + \Delta\right] = Y\left[\mathbf{y}(n-1)\right] + \Delta^T \left.\frac{\mathrm{d}Y(\mathbf{y})}{\mathrm{d}\mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}(n-1)} + \Delta^T \left.\frac{\mathrm{d}^2 Y(\mathbf{y})}{2\mathrm{d}\mathbf{y}^2}\right|_{\mathbf{y}=\mathbf{y}(n-1)} \Delta$$

Here Δ is an incremental vector of variables. Taking into account (8.10), one gets the following condition of stability:

$$\Delta^T \left.\frac{\mathrm{d}Y(\mathbf{y})}{\mathrm{d}\mathbf{y}}\right|_{\mathbf{y}=\mathbf{y}(n-1)} + \Delta^T \left.\frac{\mathrm{d}Y^2(\mathbf{y})}{2\mathrm{d}\mathbf{y}^2}\right|_{\mathbf{y}=\mathbf{y}(n-1)} \Delta < 0 \tag{8.11}$$

The iteration procedure gives the following increment at each step:

$$\Delta = K^*(n-1)\frac{\mathrm{d}Y(\mathbf{y})}{\mathrm{d}\mathbf{y}}\bigg|_{\mathbf{y}=\mathbf{y}(n-1)} \tag{8.12}$$

Putting (8.12) into (8.11), one gets after some transformations

$$\left[\frac{\mathrm{d}Y(\mathbf{y})}{\mathrm{d}\mathbf{y}}\bigg|_{\mathbf{y}=\mathbf{y}(n-1)}\right]^T\left[K^{*T}(n-1)+K^{*T}(n-1)\frac{\mathrm{d}^2Y(\mathbf{y})}{2\mathrm{d}\mathbf{y}^{-2}}\bigg|_{\mathbf{y}=\mathbf{y}(n-1)}K^*(n-1)\right]\left[\frac{\mathrm{d}Y(\mathbf{y})}{\mathrm{d}\mathbf{y}}\bigg|_{\mathbf{y}=\mathbf{y}(n-1)}\right]<0$$

Consequently, the sufficient stability condition is the negative definiteness of the following matrix:

$$G=\left[K^{*T}(n-1)+K^{*T}(n-1)\frac{\mathrm{d}^2Y(\mathbf{y})}{2\mathrm{d}\mathbf{y}^2}\bigg|_{\mathbf{y}=\mathbf{y}(n-1)}K^*(n-1)\right]$$

This matrix determines the relationship between Lagrange function parameters and parameters of matrix K^* of the search system.

8.4.7
Convergence of the Iteration Search Method under the Equality-Type Constraints

Let us consider the search process convergence in the case of quadratic function. In this case, $Y(\mathbf{a}^T\lambda) = \mathbf{a}^TA\mathbf{a} + B^T\mathbf{a} + C + \lambda^TQ\mathbf{a}$:

$$\frac{\mathrm{d}Y(\mathbf{a},\lambda)}{\mathrm{d}(\mathbf{a},\lambda)}=\begin{bmatrix}2A\mathbf{a}+B^T+Q^T\lambda\\ Q\mathbf{a}\end{bmatrix} \tag{8.13}$$

This expression can be rewritten in the form

$$\frac{\mathrm{d}Y(\mathbf{a},\lambda)}{\mathrm{d}(\mathbf{a},\lambda)}=\begin{bmatrix}2A & Q^T\\ Q & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{a}\\ \lambda\end{bmatrix}+\begin{bmatrix}B^T\\ \mathbf{0}\end{bmatrix}=\widehat{A}\begin{bmatrix}\mathbf{a}\\ \lambda\end{bmatrix}+\widehat{B}$$

In this case,

$$\mathbf{y}(n)=\mathbf{y}(n+1)+K^*(n-1)\left[\widehat{A}\mathbf{y}(n-1)+\widehat{B}\right]$$

As it was done above, the following non-recurrent expression can be written for the generalized state variable

$$\mathbf{Y}(n)=\left[Y+K^*\widehat{A}\right]^n\mathbf{y}(0)+\left[\left(Y+K^*\widehat{A}\right)^n-Y\right]\widehat{A}^{-1}\widehat{B}$$

The substitution of $\mathbf{Y}(n) = \widehat{A}^{-1}\widehat{B}$ into (8.13) shows that gradient vector of the Lagrange function becomes zero, i.e., the point $\mathbf{Y}(n) = \widehat{A}^{-1}\widehat{B}$ is the extremum point. It is sufficient to prove that

$$\lim_{n\to\infty}\left[Y + K^{*}\widehat{A}\right]^{n} = 0$$

for the convergence of the iteration procedure.

In the case of matrix $[Y + K^{*}\widehat{A}]$ nonsingularity, the last expression is equivalent to the following one:

$$\left|\mathrm{Det}\left[Y + K^{*}\widehat{A}\right]\right| < 1$$

8.5 Iteration Extremum Search Methods for Multivariable Functions under Inequality-Type Constraints

Such constraints for the neural network emerge in particular due to the limits for the variance of adjustable coefficients and have the form $q_{\mu}(\mathbf{a}) \le 0$ $(\mu = 1, \ldots, M_2)$.

The following constraints of this particular type usually emerge for the neural network:

$$\left.\begin{aligned} a_i - a_{\max} &\le 0 \\ a_{\min} - a_i &\le 0 \end{aligned}\right\} \tag{8.13a}$$

In the particular case of the neural network design based on the real physical elements, the following conditions are possible:

$$a_{\max} > 0;\ a_{\min} = 0$$

$$a_{\max} = -a_{\min} > 0$$

8.5.1 Conditions of Optimality

Conditions of optimality in this case are given by the Kune-Tacker theorem. It represents the Lagrange method generalization in the case of inequality-type constraints. According to this theorem, the optimal vector **a** corresponding to the minimum of the convex functional is the solution of the following system of equations and inequalities:

$$\frac{dY(\mathbf{a},\lambda)}{d\mathbf{a}} = \frac{dY(\mathbf{a})}{d\mathbf{a}} + Q(\mathbf{a})\lambda = 0$$

$$\begin{cases} \mathbf{q}(\mathbf{a}) + \boldsymbol{\delta} = 0 \ , & \boldsymbol{\lambda} \ge 0 \\ \boldsymbol{\lambda}^{T}\delta = 0 \ , & \boldsymbol{\delta} > 0 \end{cases} \tag{8.14}$$

The expression for matrix **Q** is preserved with exchange of M_1 by M_2. In the expression (8.14),

$$\boldsymbol{\lambda} = \left[\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3, \ldots, \boldsymbol{\lambda}_{M_2}\right] \quad , \quad \boldsymbol{\delta} = \left[\boldsymbol{\delta}_1, \boldsymbol{\delta}_2, \boldsymbol{\delta}_3, \ldots, \boldsymbol{\delta}_{M_2}\right]$$

Inequalities $\boldsymbol{\delta} \geq 0$ and $\boldsymbol{\lambda} \geq 0$ mean that all components of these vectors are non-negative. It is also assumed that there exists such a vector **a** that satisfies the inequality $q_\mu(\mathbf{a}) \leq 0$. Conditions (8.14) have the following physical sense. If some constraint is not significant for the optimal vector $\mathbf{a}_{\text{opt}}$, i.e., $q_\mu(\mathbf{a}_{\text{opt}}) < 0$ for some μ, then corresponding $\lambda_\mu = 0$. If $\lambda_\mu > 0$, then it follows from (8.14) that $\delta_\mu = q_\mu(\mathbf{a}_{\text{opt}}) = 0$.

Thus, Langrangian multipliers can be regarded as some estimations of the constraint's influence upon the optimal value of the adjustable coefficient vector. If functions $Y(\mathbf{a})$ and $q_\mu(\mathbf{a})$ ($\mu = 1, \ldots, M_2$) are convex, then the Kune-Tacker theorem gives necessary and sufficient optimality conditions.

8.5.2
Algorithm of Extremum Search in the Case of Inequality-Type Constraints

The optimality conditions (8.14) give the following system of relationships for the iteration extremum search procedure in the case of inequality-type constraints:

$$\mathbf{a}(n+1) = \mathbf{a}(n) + \left[K^*_{aa}(n)\frac{dY(\mathbf{a},\boldsymbol{\lambda})}{d\mathbf{a}} + K^*_{a\lambda}(n)\frac{dY(\mathbf{a},\boldsymbol{\lambda})}{d\boldsymbol{\lambda}}\right]_{\substack{\mathbf{a}=\mathbf{a}(n)\\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}}$$

$$\underset{\boldsymbol{\lambda}(0)\geq 0}{\boldsymbol{\lambda}(n+1)} = \max\left\{0, \boldsymbol{\lambda}(n) + \left[K^*_{\lambda a}(n)\frac{dY(\mathbf{a},\boldsymbol{\lambda})}{d\mathbf{a}} + K^*_{\lambda\lambda}(n)\frac{dY(\mathbf{a},\boldsymbol{\lambda})}{d\boldsymbol{\lambda}}\right]_{\substack{\mathbf{a}=\mathbf{a}(n)\\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}}\right\}$$

Finally one obtains

$$\mathbf{a} = (n+1) = \mathbf{a}(n) + K^*_{aa}(n)\left[\frac{dY(\mathbf{a})}{d\mathbf{a}} + Q(\mathbf{a})\boldsymbol{\lambda}\right]_{\substack{\mathbf{a}=\mathbf{a}(n)\\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}} + K^*_{a\lambda}(n)\mathbf{q}(\mathbf{a})\Big|_{\mathbf{a}=\mathbf{a}(n)}$$

$$\lambda(n+1) = \max\left\{0, \lambda(n) + K^*_{\lambda a}(n)\left[\frac{dY(\mathbf{a})}{d\mathbf{a}} + Q(\mathbf{a})\boldsymbol{\lambda}\right]_{\substack{\mathbf{a}=\mathbf{a}(n)\\ \boldsymbol{\lambda}=\boldsymbol{\lambda}(n)}} + K^*_{\lambda\lambda}(n)\mathbf{q}(\mathbf{a})\Big|_{\mathbf{a}=\mathbf{a}(n)}\right\}$$

In the particular case of constraints (8.13*a*), $\mathbf{q}_1(\mathbf{a}) = \mathbf{a} - \mathbf{a}_{\max} \leq 0$; $\mathbf{q}_2(\mathbf{a}) = \mathbf{a}_{\min} - \mathbf{a} \leq 0$;

$$\mathbf{q}(\mathbf{a}) = \begin{bmatrix} \mathbf{a} - \mathbf{a}_{\max} \\ \mathbf{a}_{\max} - \mathbf{a} \end{bmatrix}$$

$$\mathbf{Q}(\mathbf{a}) = \left[\begin{array}{ccc|ccc} 1 & 0 & 0\ldots 0 & -1 & 0 & 0\ldots 0 \\ 0 & 1 & 0\ldots 0 & 0 & -1 & 0\ldots 0 \\ 0 & 0 & 0\ldots 1 & 0 & 0 & 0\ldots 1 \end{array}\right]$$

8.6 Algorithm of Random Search of Local and Global Extrema for Multivariable Functions

The only reason for introducing randomness into the procedure of the neural network secondary optimization functional extremum search is the multi-modality of the input signal distribution. This property results in multi-extremum characteristics of the quality function of the open-loop neural network with the given structure. The random search methods are described in details in [8-2, 8-16].

The problem is to find local minimums of the multi-extremum functional of the neural network error and selection of the global minimum. Let us describe one cycle of the random search algorithm:

a The vector value of the function variables is randomly selected. This vector must be located in the region of some local extremum;
b This extremum must be found by any of the non-random search methods given above;
c The value of the found extremum and corresponding vector of variables are compared with those stored in the memory. If this extremum is absent in the memory, then it is stored too;
d The transfer to the step "a" is performed thereafter.

The results of experimental investigation of this algorithm are given in Chap. 12. The convergence rate can be in principle increased by eliminating already found regions from the random region set for the initial vector.

Let us analyze the convergence of this random search algorithm with respect to the number of extrema. Let i modes be already found ($0 \le i \le U$). The probability of occurrence in the region of these modes under their uniform distribution is i/U. The distribution of the random value ξ_i representing the number of steps from the search of the i-th mode to the search of the $(i+1)$-th mode amounts to $\xi_i = k$ with the probability

$$\left[\frac{i}{U}\right]^{k-1}\left[1-\frac{i}{U}\right] \tag{8.15}$$

The random search procedure is performed independently at each step. Let us introduce a new random value

$$\eta_j = \sum_{i=0}^{j-1} \xi_i (1 \le j \le U)$$

representing the number of steps up to the search of j modes out of U. Independent events $\xi_i = k_1, \ldots, \xi_{k-1} = k_{j-1}$, where $1 \le k \le s+1, \ldots, 1 \le k_{j-1} \le s+1$ and $k_1 + k_2 + \ldots + k_{j-1} = s + j - 1$ give in their combination such an event that $\eta_j = \xi_0 + \ldots + \xi_{j-1} = s + j$. Hence, $\xi_0 = 1$ with the unit probability. Due to the independence of ξ_j, the probability of such an event is

$$P(\xi_1 = k_1) \ldots P(\xi_{j-1} = k_{j-1})$$

According to the formula of total probability,

$$P(\eta_j = s + j) = \sum_{k_1 + \ldots + k_{j-1} = s + j - 1} P(\xi_1 = k_1) \ldots P(\xi_{j-1} = k_{j-1})$$
$$= U^{1-j-s} \frac{(U-1)!}{(U-j)!} \sum_{k'_1 + \ldots + k'_{j-1} = s (k'_r \geq 0)}^{k_r \geq 1} \prod_{i=1}^{j-1} i^{k'_i} \tag{8.16}$$

In the particular case of $j = U$,

$$P(\eta_U = s + U) = U^{1-U-s}(U-1)! \sum_{\substack{k'_1 + \ldots + k'_{U-1} = s \\ (k_r \geq r = 1, \ldots, U-1)}} \prod_{i=1}^{U-1} i^{k'_r}$$

where $P(\eta_U = s + U)$ is the probability that U modes will be found through $s + U$ steps of the random search procedure. It can be shown that the average value and variance of the number of steps required for the search of U modes are

$$\left.\begin{aligned} M\eta_U &= 1 + U \sum_{r=1}^{U-1} \frac{1}{r} \approx 1 + U\left[\ln(U-1) + 0.577\right] \\ D\eta_U &= \sum_{r=1}^{U-1} \frac{U(U-r)}{r^2} \approx 2U^2 - U\left[\ln(U-1) + 0.577\ldots\right] \end{aligned}\right\} \tag{8.16a}$$

The analysis of these expressions shows a sufficient convergence rate of this search procedure. The procedure can be generalized for the case when the region of the already found mode is excluded from the random search as it was mentioned above. The above expressions are valid in the multidimensional case.

8.7 Development of the Neural Network Adaptation Algorithms with the Use of Estimations of the Second Order Derivatives of the Secondary Optimization Functional

Let us consider the development of the neural network adaptation algorithms with the use of simultaneous estimations of the first and second order derivatives of the secondary optimization functional.

8.7.1 Development of Search Algorithms

Let us consider the problem of extremum search in the form of an equivalent problem of the root search for the following system of equations:

$$\left.\begin{aligned} &D_1(x_1, \ldots, x_N) = 0 \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ &D_N(x_1, \ldots, x_N) = 0 \end{aligned}\right\} \tag{8.17}$$

When considering the implicitly given system

$$\left.\begin{array}{l} y_1 - D_1(x_1,...,x_N) = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ y_N - D_N(x_1,...,x_N) = 0 \end{array}\right\} \tag{8.18}$$

that satisfies the Yung theorem, there exist such

$$\left.\begin{array}{l} x_1(y_1,\ldots,y_N) = F_1(y_1,\ldots,y_N) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_N(y_1,\ldots,y_N) = F_N(y_1,\ldots,y_N) \end{array}\right\} \tag{8.19}$$

that their substitution into (8.18) provides identity. Let us transform functions $F_1(y_1, \ldots, y_N), \ldots, F_N(y_1, \ldots, y_N)$ into the Taylor series with two terms:

$$\left.\begin{array}{l} F_1(0) = F_1(\mathbf{y}) - \sum\limits_{i=1}^{N} F'_{1y_i}(\mathbf{y}) y_i + \dfrac{1}{2!}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N} \dfrac{\partial^2 F_1(y)}{\partial y_i \partial y_j} y_i y_j + R_{31} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ F_N(0) = F_N(\mathbf{y}) - \sum\limits_{i=1}^{N} F'_{Ny_i}(\mathbf{y}) y_i \ldots + \dfrac{1}{2!}\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{N} \dfrac{\partial^2 F_N(y)}{\partial y_i \partial y_j} y_i y_j + R_{3N} \end{array}\right\} \tag{8.20}$$

Differentiation of (8.18) with the use of (8.19) results in the following system of equations:

$$\left.\begin{array}{c} \sum\limits_{i=1}^{N} \dfrac{\partial D_1}{\partial x_i}\dfrac{\partial x_i}{\partial y_k} = 0 \;\; ; \quad \sum\limits_{i=1}^{N} \dfrac{\partial D_k}{\partial x_i}\dfrac{\partial x_i}{\partial y_k} = 1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \sum\limits_{i=1}^{N} \dfrac{\partial D_N}{\partial x_i}\dfrac{\partial x_i}{\partial y_k} = 0 \end{array}\right\} \tag{8.21}$$

$$\begin{bmatrix} \dfrac{\partial F_1}{\partial y_1} \cdots \dfrac{\partial F_1}{\partial y_N} \\ \ldots\ldots \\ \dfrac{\partial F_N}{\partial y_1} \cdots \dfrac{\partial F_N}{\partial y_N} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial D_1}{\partial x_1} \cdots \dfrac{\partial D_1}{\partial x_N} \\ \ldots\ldots \\ \dfrac{\partial D_N}{\partial x_1} \cdots \dfrac{\partial D_N}{\partial x_N} \end{bmatrix}^{-1}$$

After differentiation of (8.21), one obtains

$$\sum_{i=1}^{N} \frac{\partial^2 x_i}{\partial y_k \partial y_1}\frac{\partial D_1}{\partial x_i} + \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial^2 D_1}{\partial x_i \partial x_j}\frac{\partial x_j}{\partial y_l}\frac{\partial x_i}{\partial y_k} = 0$$

$$\sum_{i=1}^{N} \frac{\partial^2 x_i}{\partial y_k \partial y_l}\frac{\partial D_N}{\partial x_i} + \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\partial^2 D_N}{\partial x_i \partial x_j}\frac{\partial x_j}{\partial y_l}\frac{\partial x_i}{\partial y_k} = 0$$

After multiplying of both sides of the equations by $y_k y_l$ and summation over k and l, one obtains

$$\sum_{i=1}^{N}\frac{\partial D_1}{\partial x_i}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial^2 x_i}{\partial y_k \partial y_l}y_k y_l = -\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2 D_1}{\partial x_i \partial x_j}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial x_j}{\partial y_1}\frac{\partial x_i}{\partial y_k}y_l y_k = C_1$$

.....

$$\sum_{i=1}^{N}\frac{\partial D_N}{\partial x_i}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial^2 x_i}{\partial y_l \partial y_k}y_k y_l = -\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2 D_N}{\partial x_i \partial x_j}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial x_j}{\partial y_l}\frac{\partial x_i}{\partial y_k}y_l y_k = C_N$$

This system can be rewritten in the form

$$\begin{bmatrix}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial^2 x_1}{\partial y_k \partial y_l}y_k y_l \\ \cdots\cdots\cdots \\ \sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial^2 x_N}{\partial y_k \partial y_l}y_k y_l\end{bmatrix} = -\begin{bmatrix}\frac{\partial D_1}{\partial x_1}\cdots\frac{\partial D_1}{\partial x_N} \\ \cdots\cdots\cdots \\ \frac{\partial D_N}{\partial x_1}\cdots\frac{\partial D_N}{\partial x_N}\end{bmatrix}^{-1} C$$

Taking into account that the vector

$$\alpha = \begin{bmatrix}\alpha_1 \\ . \\ . \\ . \\ \alpha_N\end{bmatrix} = \begin{bmatrix}F_1(0) \\ . \\ . \\ . \\ F_N(0)\end{bmatrix}$$

is the root of the system (8.17),

$$W^{-1} = \begin{bmatrix}\frac{\partial D_1}{\partial x_1}\cdots\frac{\partial D_1}{\partial x_N} \\ \frac{\partial D_N}{\partial x_1}\cdots\frac{\partial D_N}{\partial x_N}\end{bmatrix}; \quad x = \begin{bmatrix}x_1 \\ . \\ . \\ x_N\end{bmatrix}; \quad y_k = D_k(x)$$

$$C = \begin{bmatrix}C_1 \\ . \\ . \\ C_N\end{bmatrix} = \begin{bmatrix}\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2 D_1}{\partial x_i \partial x_j}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial x_j}{\partial y_l}\frac{\partial x_i}{\partial y_k}y_k y_l \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2 D_N}{\partial x_i \partial x_j}\sum_{k=1}^{N}\sum_{l=1}^{N}\frac{\partial x_j}{\partial y_l}\frac{\partial x_i}{\partial y_k}y_k y_l\end{bmatrix}$$

It follows from (8.20)

$$\alpha = x - W^{-1}D + \frac{1}{2!}W^{-1}C$$

Consequently, the general expression for the search algorithm of the multivariable function extremum with the second order derivatives matrix is

$$x(n+1)=x(n)-W^{-1}[x(n)]D[x(n)]+\frac{1}{2!}W^{-1}[x(n)]C[x(n)] \tag{8.22}$$

8.7.2 One-Dimensional Case

In this case, $D(x) = 0$; $F(y) = D^{-1}(x)$; $x = F[D(x)]$, ($x \in [a, b]$); $y = D[F(y)]$. If α is the root of the equation, then $\alpha = F(\mathbf{0})$. Let us transform $F(y)$ into the series

$$F(0)-F(y)=\sum_{k=1}^{r}(-1)^k\frac{F^k(y)}{k!}y^k+R_{r+1}$$

or in another form,

$$\alpha=x+\sum_{k=1}^{r}(-1)^k\frac{F^{(k)}[D(x)]}{k!}[D(x)]^k+R_{r+1}$$

The differentiation of the initial equations gives

$$F'\big[D(x)\big]D'(x)=1$$

$$F''\big[D(x)\big]D'^2(x)+F'\big[D(x)\big]D''(x)=0$$

$$F'''\big[D(x)\big]D'^3(x)+3F''\big[D(x)\big]D'(x)D''(x)+F'\big[D(x)\big]D'''(x)=0$$

One finally obtains in the case of $r = 2$

$$x(n+1)=x(n)-\frac{D'(x_n)}{D''(x_n)}+\frac{D'^2(x_n)}{2D''^3(x_n)}D'''(x_n) \tag{8.23}$$

where $D'''(x_n) = K^*(n)$.

Literature

[8-1] Tsypkin Ya Z (1966) Adaptation, learning and self-learning in automatic systems. Automatics and telemechanics1:23–62

[8-2] Rastrigin LA (1965) Random search in the problem of multi-parameter system optimization. Riga, Zinatne

[8-3] Tsypkin Ya Z (1966) The use of the stochastic approximation method for the estimation of unknown distribution density by observations. Automatics and telemechanics 3:94–96

[8-4] Deviaterikov IP, Propoj AI, Tsypkin Ya Z (1967) About recurrent learning algorithms for pattern recognition. Automatics and telemechanics 1:122–132

[8-5] Tsypkin Ya Z, Kalmans GK (1967) Recurrent self-learning algorithms. Automatics and telemechanics 5:78–87

[8-6] Tsypkin Ya Z (1968) Adaptation and learning in automatic systems. Moscow, Nauka
[8-7] Tsypkin Ya Z (1968) Optimal hybrid adaptation and learning algorithms. Automatics and telemechanics 9:96
[8-8] Tsypkin Ya Z (1970) Learning system theory foundation. Moscow, Nauka
[8-9] Tsypkin Ya Z (1970) Generalized learning algorithms. Automatics and telemechanics, pp 97–104
[8-10] Tsypkin Ya Z Learning automatic systems. Automatics and telemechanics 4:55–71
[8-11] Zabrejko PP, Krasnoselski MA, Tsypkin Ya Z (1970) About optimal and sub-optimal learning algorithms. Automatics and telemechanics 10:91–98
[8-12] Galushkin AI (1970) Analysis of one extremum search iteration method. Automatics and computer engineering, AN Latv.SSR, pp 38–40
[8-13] Tsypkin Ya Z (1971) Smoothed randomized functionals and algorithms in the adaptation and learning theory. Automatics and telemechanics 8:29–50
[8-14] Galushkin AI, Tiukhov BP, Chigrinov VG (1971) About convergence of one random search method for the search of local and global extrema of multivariable function. MIEM Proc. 23:205–209
[8-15] Galushkin AI, Scmid AV (1971) Iteration methods for the search of extrema of multivariable functions under the equality-type constraints. Automatics and computer engineering, AN Latv. SSR, 4:88–91
[8-16] Rastrigin LA (1971) Random search with linear tactics. Riga, Zinatne
[8-17] Tsypkin Ya Z (1972) Dynamic adaptation algorithms. Automatics and telemechanics 1:68–76
[8-18] Tsypkin Ya Z (1972) Learning algorithms for recognition under the transient conditions. Information transmission problems 3:94–102
[8-19] Poliak BT, Tsypkin Ya Z (1973) Pseudo-gradient adaptation and learning algorithms. Automatics and telemechanics 3:45–68
[8-20] Tsypkin Ya Z, Kaplinski AI, Krasnenker AS (1973) Methods of local improvement in the stochastic optimization problems. Izv. AN SSSR, ser. Technology cybernetics, pp 3–11
[8-21] Tsypkin Ya Z (1976) Adaptive methods for solution selection under indeterminateness conditions. Automatics and telemechanics 3:78–91
[8-22] Tsypkin Ya Z (1976) Optimization under the indeterminateness conditions. Dokl. AN SSSR 228(6):1306–1309
[8-23] Tsypkin Ya Z (1977) Stabilization and regularization of optimal solution estimation under the indeterminateness conditions. Dokl. AN SSSR 236(2):304–306
[8-24] Tsypkin Ya Z (1977) About some properties of the random search. Automatics and telemechanics 11:89–94
[8-25] Poliak BT, Tsypkin Ya Z (1980) Optimal pseudo-gradient stochastic optimization methods. Dokl. AN SSSR 250(5):1084–1087
[8-26] Poliak BT, Tsypkin Ya Z (1980) Optimal pseudo-gradient adaptation algorithms. Automatics and telemechanics 8:74–84
[8-27] Poliak BT, Tsypkin Ya Z (1980) Robust pseudo-gradient adaptation algorithms. Automatics and telemechanics 10:91–97
[8-28] Tsypkin Ya Z, Pozniak AS (1981) Optimal search algorithms of stochastic optimization. Dokl. AN SSSR 260(3):550–553
[8-29] Ivakhnenko AG (1969) Self-learning recognition and automatic control systems. Kiev, Tekhnika

Part III Adaptive Neural Networks

Chapter 9 Neural Network Adjustment Algorithms

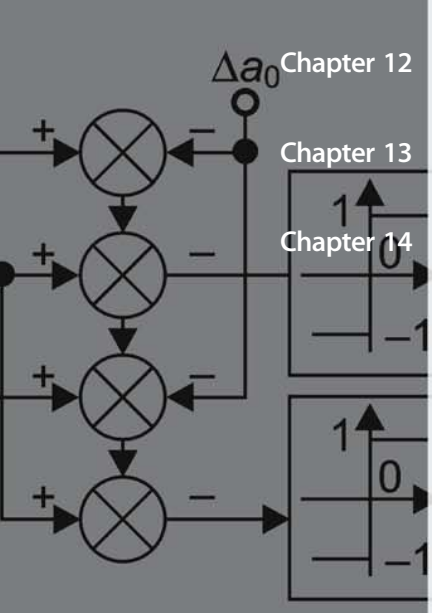
Chapter 10 Adjustment of Continuum Neural Networks

Chapter 11 Selection of Initial Conditions During Neural Network Adjustment – Typical Neural Network Input Signals

Chapter 12 Analysis of Closed-Loop Multilayer Neural Networks

Chapter 13 Synthesis of Multilayer Neural Networks with Flexible Structure

Chapter 14 Informative Feature Selection in Multilayer Neural Networks



Chapter 9

Neural Network Adjustment Algorithms

9.1 Problem Statement

It was mentioned in Chap. 7 that secondary optimization functional selection is performed on the basis of the given general input signal characteristics, primary optimization criterion and open-loop neural network structure.

The closed-loop neural network represents an open-loop neural network with included adjustment unit. Development of the closed-loop neural network is performed on the basis of a selected secondary optimization criterion and on the method of extremum search of the given functional [9-1 to 9-4]. Synthesis of the unit for the calculation of parameters of the neural network quality functional required for the iteration search organization is performed. The main problem in this case consists in the estimation of the secondary optimization functional gradient vector. There are two possibilities to solve this problem. It can be done by the search procedure on the basis of analysis of search oscillation results or by the determination of gradient vector estimation in the form of analytical expression.

In the first case, the search neural network is used, and in the second one – the analytical neural network is used. The first case is preferable but not always possible. If the system does not provide the possibility to select the signal characterizing the optimization functional gradient, then one must use the search oscillation methods.

We consider below recognition systems of different types: a neuron with two solutions for two pattern classes, a neuron with K_p solutions for K pattern classes, a neuron with a solution continuum and continuum of pattern classes, multilayer neural networks consisting of neurons with a solution continuum and the existence or absence of constraints upon the adjustable coefficients, multilayer neural networks with N^*-dimensional signals $\varepsilon(n)$ and $y(n)$, and multilayer neural networks with cross and backward connections.

Methods of neural network design can be easily generalized for the case of transient patterns when the implementation of the gradient vector is the realization of a nonstationary multidimensional random process. This gradient property determines the method of multidimensional filter design in the neural network adjustment unit.

A special attention requires the problem of multilayer neural network design in the learning mode and in the mode with arbitrary teacher qualification. Methods of closed-loop neural network design in this case are the same as in the learning mode.

Design of a neural network adjustment algorithm with a closed cycle is performed by putting the expression for the estimation of the functional gradient vector into the corresponding expression for the search procedure.

9.2 Neuron with Two-Solution Continuums

Four secondary optimization functionals $|\alpha_{1a}|$, $|\alpha_{2a}|$, $|\alpha_{1g}|$, and $|\alpha_{2a}|$ are considered below for the neuron with two solutions. The expression for the estimation of the absolute value for the first moment of the analogous error has the form

$$\left|\overline{x_a(n)}^{m_n}\right| = \left|\overline{\varepsilon(n)}^{m_n} - \sum_{i=0}^{N} \overline{a_i x_i(n)}^{m_n}\right| \quad , \quad (x_0 = -1)$$

Consequently,

$$\frac{\partial\left|\overline{x_a(n)}^{m_n}\right|}{\partial a_i} = -\operatorname{sign}\left[\overline{x_a(n)}^{m_n}\right]\overline{x_i(n)}^{m_n} \quad , \quad i = 0,\ldots,N$$

The recurrent expression representing a basis for the design of a neuron that is adjustable through the closed cycle has the following form:

$$\mathbf{a}(n+1) = \mathbf{a}(n) - K^* \operatorname{sign}\left[\overline{x_a(n)}^{m_n}\right]\overline{\mathbf{x}(n)}^{m_n} \tag{9.1}$$

The choice of parameters of matrix K^* is the final goal of the analysis and synthesis of closed-loop neural networks. Some constraints can be imposed upon this matrix even at the given stage. In [9-5, 9-6], these constraints are determined for the stochastic approximation method. One can require the convergence of the iteration procedure to its extremum $|\alpha_{1a}|$ at each step, i.e., the validity of the following conditions (in a one-dimensional case):

$$\overline{\varepsilon(n)}^{m_n} - \overline{x(n)}^{m_n} a_1(n+1) + a_0(n+1) = 0$$

One of the possible sets of matrix K^* elements is

$$K_{11}^* = -\left|\overline{x_a(n)}^{m_n}\right| \quad , \quad K_{21}^* = K_{12}^* = K_{22}^* = 0$$

The choice of m_n in the expression

$$\overline{x_a(n)}^{m_n} = \frac{1}{m_n} \sum_{i=n-m_n}^{m_n} x_a(i)$$

is also a problem of analysis and synthesis of the closed-loop neural networks. The increase of m_n results in the increase of the noise level for the measurement of the secondary optimization functional gradient. But it decreases the delay in the closed-cycle adjustment circuit.

In the case of minimization of the second order moment of the analogous error,

$$\overline{x_a^2(n)}^{m_n} = 1 + \left[\overline{\sum_{i=0}^{N} a_i x_i(n)}^{m_{n2}}\right] - 2\varepsilon(n)\overline{\sum_{i=0}^{N} a_i x_i(n)}^{m_n}$$

Consequently,

$$\frac{\partial \overline{x_a^2(n)}^{m_n}}{\partial a_i} = -2\,\overline{x_a(n)\cdot x_i(n)}^{m_n} \quad , \quad i = 0,\ldots,N$$

$$\mathbf{a}(n+1) = \mathbf{a}(n) - 2K^{*}\,\overline{x_a(n)\mathbf{x}(n)}^{m_n}$$

In the case of minimization of the first order moment of the neural network discrete error,

$$\left|\overline{x_g(n)}^{m_n}\right| = \left|\overline{\varepsilon(n)}^{m_n} - \operatorname{sign}\left[\overline{\sum_{i=0}^{N} a_i x_i(n)}^{m_n}\right]\right|$$

$$\frac{\partial\left|\overline{x_g(n)}^{m_n}\right|}{\partial a_i} = -\operatorname{sign}\left[\overline{x_g(n)}\right]^{m_n}\overline{\frac{\partial}{\partial a_i}\operatorname{sign} g(n)}^{m_n}$$

For the extremum search, one can use the information about signs and values of the first and second derivatives, etc.

The value of the first derivative in this case cannot be determined, and one must use the information about its sign. Taking into account that

$$\frac{\partial}{\partial a_i}\operatorname{sign}\sum_{i=0}^{N} a_i x_i(n) = \lim_{B\to\infty}\frac{2}{\pi}\frac{\partial}{\partial a_i}\operatorname{arctg} B\sum_{i=0}^{N} a_i x_i(n) = \lim_{B\to\infty}\frac{2}{\pi}\frac{Bx_i(n)}{1+B^2 g^2(n)}$$

then

$$\operatorname{sign}\left[\frac{\partial}{\partial a_i}\operatorname{sign} g(n)\right]\operatorname{sign}\left[\frac{2}{\pi}x_i(n)\lim_{B\to\infty}\frac{B}{1+B^2 g^2(n)}\right] = \operatorname{sign} x_i(n)$$

Consequently,

$$\frac{\partial\left|x_g(n)\right|}{\partial a_i} = \operatorname{sign}\left[x_g(n)\right]\operatorname{sign} x_i(n) \quad , \quad i = 0,\ldots,N \tag{9.2}$$

Here and below the latter expression is also conditionally termed as an estimation of a gradient vector, though in principle this is a pseudo-gradient resulting from the exchange of the derivative $\partial y/\partial a_i$ by the sign of the derivative.

In this case, there is no possibility to create the closed-cycle adjustment algorithm satisfying criterion of minimum $|\alpha_{1g}|$ under the arbitrary value of memory m_n for the gradient estimation filter. To show this, let us present the measured values of the second-

ary optimization functional gradient in the form of some random process. In the general case, including the criterion of minimum α_{2g}, the measured gradient at the current step can be conditionally represented in the form of the product of two multipliers $x_1(n)x_2(n)$. One of the multipliers, for example $\partial[\operatorname{sign} g(n)]/\partial a_i$ can not be calculated directly through the neural network signals. Only its sign can be determined in this way. The exchange in the expression for the gradient under the arbitrary value of m_n results in the loss of the possibility to determine the sign of gradient estimation because in the general case,

$$\operatorname{sign}\overline{x_1(n)x_2(n)}^{m_n} \neq \operatorname{sign}\overline{x_1(n)\operatorname{sign}x_2(n)}^{m_n}$$

Consequently, the design of analytical adjustment algorithms for the neural network with two solutions and closed-cycle secondary optimization functional for discrete error can be performed only if $m_n = 1$. Otherwise, if $m_n > 1$, then the search adjustment procedure must be developed. In any case, the adjustment procedure for the estimation of $\partial y/\partial a$ in the expression of the secondary optimization gradient must be introduced.

Expression (9.2) represents a basis for implementation of a corresponding closed-loop neural network. In the case of the neural network with α_{2g} minimization,

$$\overline{\frac{\partial x_g^2(n)}{\partial a_i}}^{m_n} = \overline{-2x_g(n)\operatorname{sign}x_i(n)}^{m_n}$$

It is evident that the adjustment algorithm with minimization criteria of $|\alpha_{1g}|$ and α_{2g} are the same if $m_n = 1$.

In the case of a neuron with a solution continuum,

$$y(n) = F[g(n)] = F\left[\sum_{i=0}^{N} a_i x_i(n)\right]$$

$$x_g(n) = \varepsilon(n)F\left[\sum_{i=0}^{N} a_i x_i(n)\right]$$

In the case of minimization of $|\alpha_{1g}|$ and α_{2g} respectively,

$$\frac{\partial\left|\overline{x_g(n)}^{m_n}\right|}{\partial a_i} = \operatorname{sign}\left[\overline{x_g(n)}^{m_n}\right]\overline{\frac{\mathrm{d}F(g)}{\mathrm{d}g}x_i(n)}^{m_n}$$

$$\overline{\frac{\partial x_g^2(n)}{\partial a_i}}^{m_n} = \overline{-2x_g(n)\frac{\mathrm{d}F(g)}{\mathrm{d}g}}^{m_n} x_i(n)$$

Recurrent algorithms for the development of the closed-loop neural network in the considered cases are

$$\mathbf{a}(n+1) = \mathbf{a}(n) + K^* \operatorname{sign}\left[\overline{x_g(n)}^{m_n}\right]\overline{\frac{\mathrm{d}F(g)}{\mathrm{d}g}\mathbf{x}(n)}^{m_n}$$

$$\mathbf{a}(n+1)=\mathbf{a}(n)+2K_1^*\,\overline{x_g(n)\frac{\mathrm{d}F(g)}{\mathrm{d}g}\mathbf{x}(n)}^{m_n}$$

In the particular case of $F(g) = 2/\pi \operatorname{arctg} Bg$

$$\mathbf{a}(n+1)=\mathbf{a}(n)+K^*\operatorname{sign}\left[\overline{x_g(n)}^{m_n}\right]\frac{2B}{\pi}\frac{\mathbf{x}(n)}{1+B^2g^2(n)}$$

$$\mathbf{a}(n+1)=\mathbf{a}(n)+K^*\frac{4B}{\pi}\frac{\overline{\mathbf{x}(n)x_g(n)}^{m_n}}{1+B^2g^2(n)}$$

9.3 Two-Layer Neural Networks

Let us consider a two-layer neural network with the neurons with full connections adjustable by a closed cycle. In this case,

$$y(n)=F[g(n)]=F\left[\sum_{j=0}^{H_1}a_jy(n)\right]=F\left[\sum_{j=0}^{H_1}a_jF\left[g_i(n)\right](n)\right]=F\left[\sum_{j=0}^{H_1}a_jF\left[\sum_{i=0}^{N}a_{ij}x_i(n)\right]\right]$$

Here

$$x_a(n)=\varepsilon(n)-g(n);\quad x_g(n)=\varepsilon(n)-y(n)$$

Table 9.1. Expressions for the adjustable coefficients of the neuron in the first and second layers

Secondary optimization functional	$\frac{\partial(\cdot)}{\partial a_j}$	$\frac{\partial(\cdot)}{\partial a_{ij}}$
$\lvert\alpha_{1a}\rvert\,,\ \overline{\lvert x_a(n)\rvert}^{m_n}$	$-\operatorname{sign}\overline{[x_a(n)]}^{m_n}\,\overline{y(n)}^{m_n}$	$-\operatorname{sign}\overline{[x_a(n)]}^{m_n}\,\overline{a_j\frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j}x_i(n)}^{m_n}$
$\alpha_{2a}\,,\ \overline{x_a^2(n)}^{m_n}$	$-\overline{2x_a(n)y(n)}^{m_n}$	$-\overline{2a_jx_a(n)\frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j}x_i(n)}^{m_n}$
$\lvert\alpha_{1g}\rvert\,,\ \overline{\lvert x_g(n)\rvert}^{m_n}$	$-\operatorname{sign}\overline{[x_g(n)]}^{m_n}\,\overline{\frac{\mathrm{d}F(g)}{\mathrm{d}g}y(n)}^{m_n}$	$-\operatorname{sign}\overline{[x_g(n)]}^{m_n}\,\overline{a_j\frac{\mathrm{d}F(g)}{\mathrm{d}g}\frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j}x_i(n)}^{m_n}$
$\lvert\alpha_{2g}\rvert\,,\ \overline{x_g^2(n)}^{m_n}$	$-\overline{2x_g(n)\frac{\mathrm{d}F(g)}{\mathrm{d}g}y(n)}^{m_n}$	$-\overline{2a_jx_a(n)\frac{\mathrm{d}F(g)}{\mathrm{d}g}\frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j}x_i(n)}^{m_n}$

The main problem is the derivation of the expressions for the secondary optimization functional gradient through the output and intermediate neural network signals. Table 9.1 shows these expressions for the adjustable coefficients of the neuron in the first and second layers.

Tables 9.2 and 9.3 show expressions for the secondary optimization functional gradients in the case of $F(g) = \text{sign}(g)$ and $F(g) = (2/\pi)\text{arctg}\, Bg$.

Let us mention some features of the learning methods of the two-layer neural network with one layer of nonlinear random connections. Rosenblatt called such a neural network a three-layer perceptron. Its structure was described in Chap. 2. The sharp decrease of neuron inputs in the first layer and random connections of this layer with

Table 9.2. Expressions for the secondary optimization functional gradients in the case of $F(g) = \text{sign}(g)$

Secondary optimization functional	$\frac{\partial(\cdot)}{\partial a_j}$	$\frac{\partial(\cdot)}{\partial a_{ij}}$
$\lvert\alpha_{1a}\rvert,\ \overline{\lvert x_a(n)\rvert}^{m_n}$	$-\text{sign}\overline{[x_a(n)]}^{m_n}\,\overline{y(n)}^{m_n}$	$-a_j\,\text{sign}\overline{[x_a(n)]}^{m_n}\,\overline{\text{sign}\,x_i(n)}^{m_n}$
$\alpha_{2a},\ \overline{x_a^2(n)}^{m_n}$	$-\overline{2x_a(n)y(n)}^{m_n}$	$-\overline{2a_j x_a(n)\text{sign}\,x_i(n)}^{m_n}$
$\lvert\alpha_{1g}\rvert,\ \lvert x_g(n)\rvert$	$-\text{sign}[x_g(n)]y(n)$	$-\text{sign}[x_g(n)]\text{sign}\,a_j\,\text{sign}\,x_i(n)$
$\alpha_{2g},\ x_g^2(n)$	$-2x_g(n)y(n)$	$-2\text{sign}\,a_j x_g(n)\text{sign}\,x_i(n)$

Table 9.3. Expressions for the secondary optimization functional gradients in the case of $F(g) = (2/\pi)\ \text{arctg}\, Bg$

Secondary optimization functional	$\frac{\partial(\cdot)}{\partial a_j}$	$\frac{\partial(\cdot)}{\partial a_{ij}}$
$\lvert\alpha_{1a}\rvert,\ \overline{\lvert x_a(n)\rvert}^{m_n}$	$-\text{sign}\overline{[x_a(n)]}^{m_n}\,\overline{y(n)}^{m_n}$	$-\text{sign}\overline{[x_a(n)]}^{m_n}\,a_j\frac{2}{\pi}\left[\frac{\overline{Bx_i(n)}^{m_n}}{1-B^2g^2(n)}\right]$
$\alpha_{2a},\ \overline{x_a^2(n)}^{m_n}$	$-\overline{2x_a(n)y(n)}^{m_n}$	$-2a_j\frac{2}{\pi}x_a(n)\left[\frac{\overline{Bx_i(n)}^{m_n}}{1+B^2g^2(n)}\right]$
$\lvert\alpha_{1g}\rvert,\ \overline{\lvert x_g(n)\rvert}^{m_n}$	$-\text{sign}\overline{[x_g(n)]}^{m_n}\,\frac{2}{\pi}\frac{\overline{By(n)}^{m_n}}{\left[1+B^2g^2(n)\right]}$	$-\text{sign}\overline{[x_g(n)]}^{m_n}\,\frac{4}{\pi^2}a_j\left[\frac{\overline{B^2x_i(n)}^{m_n}}{\left[1+B^2g^2(n)\right]\left[1+B^2g_j^2(n)\right]}\right]$
$\lvert\alpha_{2g}\rvert,\ \overline{x_g^2(n)}^{m_n}$	$-2x_g(n)\frac{2}{\pi}\left[\frac{\overline{By(n)}^{m_n}}{1+B^2g^2(n)}\right]$	$-\frac{8}{\pi^2}a_j\left[\frac{\overline{B^2x_i(n)x_g(n)}^{m_n}}{\left[1+B^2g^2(n)\right]\left[1+B^2g_j^2(n)\right]}\right]$

the input space of the neural network result in the requirement to increase the number of neurons in the first layer.

In this case,

$$y(n)=F\left[\sum_{j=0}^{H_1} a_j F\left[\sum_{i_j=0}^{N_1} a_{i_j} x_{i_j}(n)\right]\right]$$

The random connections are fixed at the stage of adjustment. Only connection coefficients must be adjustable. The adjustment algorithm for neurons of the first layer has the form (for minimum α_{2g} criterion)

$$\frac{\overline{\partial x_g^2(n)}^{m_n}}{\partial a_{i_j}}=-2a_j x_g(n)\frac{\overline{\mathrm{d}F(g)}}{\mathrm{d}g}\frac{\mathrm{d}F(g_i)}{\mathrm{d}g_i}x_{i_j}(n)$$

9.4 Multilayer Neural Networks with Solution Continuum Neurons

The neural network in this case has H_j neurons in each j-th layer ($j=1,\ldots,W$). The expression for the output signal of such a neural network has the form (2.2). Let us find partial derivatives of $y(n)$ and $g(n)$ with respect to coefficients $a_{h_{W-j+1},W-j}$:

$$\frac{\partial y(n)}{\partial a_{h_{W-j+1},h_{W-j}}}=\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{j+2}a_{h_{W-\eta},h_{w-\eta-1}}\prod_{\nu=0}^{j+1}\frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}}x_{h_{W-j}}^{W-j}(n) \tag{9.3}$$

$$\frac{\partial g(n)}{\partial g_{h_{W-j+1},h_{W-j}}}=\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{j+2}a_{h_{W-\eta},h_{W-\eta-1}}\prod_{\nu=1}^{j+1}\frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}}x_{h_{W-j}}^{W-j}(n) \tag{9.4}$$

Tables 9.4 and 9.5 show expressions for the estimation of the secondary optimization functional gradients in the case of arbitrary F and $F=\mathrm{sign}(g)$.

In this case,

$$F(g)=\mathrm{sign}(g)\ ,\quad \mathrm{sign}\,x_{h_{W-j}}^{W-j}=x_{h_{W-j}}^{W-j}$$

for all $j\neq W$, and it significantly simplifies the expressions for the gradients.

Table 9.4. Expressions for the estimation of the secondary optimization functional gradients in the case of arbitrary F

$\lvert\alpha_{1a}\rvert$	$-\operatorname{sign}\overline{[x_a(n)]}^{m_n}\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{j+2}a_{h_W-\eta,h_{W-\eta-1}}\overline{\prod_{\nu=1}^{j+1}\frac{dF\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{dg_{h_{W-\nu}}^{W-\nu}}}^{m_n}x_{h_{W-j}}^{W-j}(n)$
α_{2a}	$-2\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{j+2}a_{h_W-\eta,h_{W-\eta-1}}x_a(n)\overline{\prod_{\nu=1}^{j+1}\frac{dF\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{dg_{h_{W-\nu}}^{W-\nu}}}^{m_n}x_{h_{W-j}}^{W-j}(n)$
$\lvert\alpha_{1g}\rvert$	$-\operatorname{sign}\overline{[x_g(n)]}^{m_n}\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{j+2}a_{h_W-\eta,h_{W-\eta-1}}\overline{\prod_{\nu=0}^{j+1}\frac{dF\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{dg_{h_{W-\nu}}^{W-\nu}}}^{m_n}x_{h_{W-j}}^{W-j}(n)$
α_{2g}	$-2\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{j+2}a_{h_W-\eta,h_{W-\eta-1}}x_g(n)\overline{\prod_{\nu=0}^{j+1}\frac{dF\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{dg_{h_{W-\nu}}^{W-\nu}}}^{m_n}x_{h_{W-j}}^{W-j}(n)$

Table 9.5. Expressions for the estimation of the secondary optimization functional gradients in the case of $F=\operatorname{sign}(g)$

$\lvert\alpha_{1a}\rvert$	$-\operatorname{sign}\overline{[x_a(n)]}^{m_n}\sum_{h_{W-1}=1}^{H_{W-1}}a_{h_W,h_{W-1}}\overline{\operatorname{sign}\left[\prod_{\eta=2}^{j-1}\sum_{h_{W-\eta}=1}^{H_{W-\eta}}a_{h_{W-\eta+1},h_{W-\eta}}x_{h_{W-j}}^{W-j}(n)\right]}^{m_n}$
α_{2a}	$-2\sum_{h_{W-1}=1}^{H_{W-1}}a_{h_W,h_{W-1}}x_a(n)\overline{\operatorname{sign}\left[\prod_{\eta=2}^{j-1}\sum_{h_{W-\eta}=1}^{H_{W-\eta}}a_{h_{W-\eta+1},h_{W-\eta}}x_{h_{W-j}}^{W-j}(n)\right]}^{m_n}$
$\lvert\alpha_{1g}\rvert$	$-\operatorname{sign}\left[x_g(n)\right]\operatorname{sign}\left[\prod_{\eta=1}^{j-1}\sum_{h_{W-\eta}=1}^{H_{W-\eta}}a_{h_{W-\eta+1},h_{W-\eta}}x_{h_{W-j}}^{W-j}(n)\right]$
α_{2g}	$-2x_g(n)\operatorname{sign}\left[\prod_{\eta=1}^{j-1}\sum_{h_{W-\eta}=1}^{H_{W-\eta}}a_{h_{W-\eta+1},h_{W-\eta}}x_{h_{W-j}}^{W-j}(n)\right]$

9.5 Design of Neural Networks with Closed Cycle Adjustment under Constraints upon Variables

Let us consider the equality-type and inequality-type constraints upon the adjustable coefficients of multilayer neural networks that were represented in Chap. 8. Such neural networks are characterized by constraints upon the assemblage of coefficients of all neural networks, upon the assemblage of coefficients of each separate layer, and upon the assemblage of each separate neuron of the neural network.

Consequently, one obtains for a two-layer neural network

$$\sum_{j=0}^{H_1}\left(a_j+\sum_{i=0}^{N}a_{ij}\right)=\alpha \tag{9.5a}$$

$$\sum_{j=0}^{H_1}\sum_{i=0}^{N}a_{ij}=\alpha_1 \;\;;\;\; \sum_{j=0}^{H_1}a_j=\alpha_2 \tag{9.5b}$$

$$\sum_{i=0}^{N}a_{ij}-\alpha_j=0 \;\;;\;\; \sum_{j=0}^{H_1}a_j-\alpha=0 \;\;;\;\; j=0,\ldots,H_1 \tag{9.5c}$$

The inequality-type constraints have the form close to that represented in Sect. 8.5.

Neural network in the form of a neuron. In the case of the $|\alpha_{1g}|$ minimization criterion and constraints (9.5a), the system (9.1) has the following form:

$$\begin{bmatrix}\mathbf{a}(n+1)\\ \lambda(n+1)\end{bmatrix}=\begin{bmatrix}\mathbf{a}(n)\\ \lambda(n)\end{bmatrix}+K^{*}\begin{bmatrix}-\operatorname{sign}[\overline{x_a(n)}]^{m_n}\overline{\mathbf{x}(n)}^{m_n}+\mathbf{l}\boldsymbol{\lambda}(n)\\ \sum_{j=0}^{N}a_i(n)-\alpha\end{bmatrix}$$

In the case of the $|\alpha_{1g}|$ minimization criterion and inequality-type constraints (Chap. 8) upon the adjustable coefficients, the recurrent relationship for the closed-loop neural network design has the form

$$\mathbf{a}(n+1)=\mathbf{a}(n)+K^{*}_{aa}(n)\left\{-\operatorname{sign}\left[x_a(n)\right]\operatorname{sign}\mathbf{x}(n)+\begin{bmatrix}\lambda_0-\lambda_{N+1}\\ \ldots\\ \lambda_N-\lambda_{2(N+1)}\end{bmatrix}\right\}+K^{*}_{a\lambda}(n)\begin{bmatrix}\mathbf{a}(n)-\mathbf{a}_{\max}\\ \ldots\\ \mathbf{a}_{\min}-\mathbf{a}(n)\end{bmatrix}$$

$$\lambda(n+1)=\max\left\{\begin{array}{l}0,\lambda(n)+K^{*}_{\lambda a}(n)\left\{-\operatorname{sign}\left[x_g(n)\right]\operatorname{sign}\mathbf{x}(n)+\begin{bmatrix}\lambda_0-\lambda_{N+1}\\ \ldots\\ \lambda_N-\lambda_{2(N+1)}\end{bmatrix}\right\}\\ +K^{*}_{\lambda\lambda}(n)\begin{bmatrix}\mathbf{a}(n)-\mathbf{a}_{\max}\\ \ldots\\ \mathbf{a}_{\min}-\mathbf{a}(n)\end{bmatrix}\end{array}\right\}$$

Two-layer neural network. Let us consider the case of constraints upon the coefficients of the multilayer neural network. The basic recurrent expressions for the closed-loop neural network design are given below.

In the case of equality-type constraints, *(a)* the second layer, *(b)* the first layer:

a

$$\mathbf{a}'(n+1)=\mathbf{a}'(n)+K^{*}_{a'a'}(n)\left[\overline{x_g(n)\left.\frac{\mathrm{d}F(g)}{\mathrm{d}g}\right|_{g=g(n)} x_k(n)}^{m_n}+1\boldsymbol{\lambda}'(n)\right]$$

$$+K^{*}_{a'\lambda'}(n)\left[\sum_{j=0}^{H_1}a_j(n)-\alpha\right]$$

$$\lambda'(n+1)=\lambda'(n)+K^{*}_{\lambda'a'}(n)\left[\overline{x_g(n)\left.\frac{\mathrm{d}F(g)}{\mathrm{d}g}\right|_{g=g(n)} x_k(n)}^{m_n}+1\boldsymbol{\lambda}'(n)\right]$$

$$+K^{*}_{\lambda'\lambda'}(n)\left[\sum_{j=0}^{H_1}a_j(n)-\alpha\right]$$

b

$$\mathbf{a}_j(n+1)=\mathbf{a}_j(n)+K^{*}_{a_ja_j}(n)\left[\overline{-2a_j(n)x_g(n)\left.\frac{\mathrm{d}F(g)}{\mathrm{d}g}\right|_{g=g(n)}\left.\frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j}\right|_{g_j=g_j(n)}\mathbf{x}(n)}^{m_n}+\lambda_j(n)\mathbf{l}\right]$$

$$+K^{*}_{a_j\lambda_j}(n)\left[\sum_{i=0}^{N}a_{ij}(n)-\alpha\right]+1\lambda_j(n)$$

$$\lambda_j(n+1)=\lambda_j(n)+K^{*}_{\lambda_ja_j}(n)\left[\overline{-2a_j(n)x_g(n)\left.\frac{\mathrm{d}F(g)}{\mathrm{d}g}\right|_{g=g(n)}}^{m_n}\mathbf{x}(n)+\lambda_j(n)\mathbf{l}\right]$$

$$+K^{*}_{\lambda_j\lambda_j}(n)\left[\sum_{i=0}^{N}a_{ij}(n)-\alpha\right]$$

In the case of inequality-type constraints, *(a)* the second layer, *(b)* the first layer:

a

$$\mathbf{a}'(n+1)=\mathbf{a}'(n)+K^{*}_{a'a'}(n)\left[\overline{x_g(n)\left.\frac{\mathrm{d}F(g)}{\mathrm{d}g}\right|_{g=g(n)} x_k(n)}^{m_n}+Q\boldsymbol{\lambda}(n)\right]+K^{*}_{a\lambda}(n)\,\mathbf{q}\left[\mathbf{a}'(n)\right]$$

$$\boldsymbol{\lambda}(n+1)=\max\left\{0,\boldsymbol{\lambda}(n)+K^{*}_{\lambda a'}(n)\left[\overline{x_g(n)\left.\frac{\mathrm{d}F(g)}{\mathrm{d}g}\right|_{g=g(n)} x_k(n)}^{m_n}+Q\boldsymbol{\lambda}(n)\right]+K^{*}_{\lambda\lambda}(n)\,\mathbf{q}\left[\mathbf{a}'(n)\right]\right\}$$

b

$$\mathbf{a}_i(n+1)=\mathbf{a}_j(n)+K^*_{a_j\lambda_j}(n)\left[\overline{-2a_j(n)x_g(n)\left.\frac{dF(g)}{dg}\right|_{g=g(n)}\left.\frac{dF(g_j)}{dg_j}\right|_{g_j=g_j(n)}\mathbf{x}(n)}^{m_n}+Q\lambda_j(n)\right]$$
$$+K^*_{a_j\lambda_j}(n)\,\mathbf{q}\left[a_j(n)\right]$$

$$\boldsymbol{\lambda}_j(n+1)=\max\left\{\begin{array}{l}0,\lambda_j(n)\\ +K^*_{\lambda_j a_j}(n)\left[\overline{-2a_j(n)x_g(n)\left.\frac{dF(g)}{dg}\right|_{g=g(n)}\left.\frac{dF(g_j)}{dg_j}\right|_{g_j=g_j(n)}\mathbf{x}(n)}^{m_n}+Q\lambda_j(n)\right]\\ +K^*_{\lambda_j\lambda_j}(n)\,\mathbf{q}\left[a_j(n)\right]\end{array}\right\}$$

Here Q and q are defined in the same way as in Chap. 8.

These algorithms can be easily generalized for the case of an arbitrary number of layers and particular arbitrary forms of constraints.

9.6 Implementation of Primary Optimization Criteria for Neurons with Two Solutions

Let us consider the minimum average risk function criterion. The expression for the transformed discrete error can be represented in the form

$$x'_g=(\varepsilon-x_k)\frac{1}{4}[(-2A+C)(\varepsilon-1)+(2B+C)(\varepsilon+1)]+\frac{1}{2}(\varepsilon+x_k)l\varepsilon$$

The gradient required for the closed-loop neural network design is

$$\frac{\partial x'^2_g}{\partial a_i}=2x'_g(n)\frac{\partial x'_g}{\partial a_i}$$

or in another form,

$$\begin{aligned}\frac{\partial x'^2_g}{\partial a_i}=&\left\{(\varepsilon-y)\frac{1}{4}[(-2A+C)(\varepsilon-1)+(2B+C)(\varepsilon+1)]+\frac{1}{2}(\varepsilon+y)l\varepsilon\right\}\\ &\times 2\operatorname{sign}x_i\left\{\frac{1}{2}l\varepsilon-\frac{1}{4}[(-2A+l)(\varepsilon-1)+(2B+C)(\varepsilon+1)]\right\}\end{aligned}\tag{9.6}$$

The values A, B, and C are determined by (7.13) and (7.14). In the case of (7.14),

$$\frac{\partial x'^2_g}{\partial a_i}=2x_g\operatorname{sign}x_i\left\{\frac{1}{8}\left[(l_{12}-l_{11})(\varepsilon-1)+(l_{21}-l_{22})(\varepsilon+1)\right]\right\}\tag{9.6a}$$

In the case of (7.14a), one obtains for the transformed discrete error

$$x'_g = \frac{1}{4}(\varepsilon + y)\left[Z_{11}(\varepsilon - 1) + Z_{22}(\varepsilon + 1)\right] + (\varepsilon - y)\left[Z_{12}(\varepsilon - 1) + Z_{21}(\varepsilon + 1)\right] \tag{9.7}$$

After corresponding transformations, we get the expression

$$\begin{aligned}\frac{\partial x'^2_g}{\partial a_i} = \frac{1}{4}\operatorname{sign} x_i \Big\{ &(1-\varepsilon)\left[(Z_{12}^2 - Z_{11}^2)x_g + y(Z_{11} - Z_{12})2Z_{11}\right] \\ &+ (1+\varepsilon)\left[(Z_{21}^2 - Z_{22}^2)x_g + y(Z_{22} - Z_{21})2Z_{22}\right]\Big\}\end{aligned} \tag{9.8}$$

that coincides with the above obtained results at $Z_{11} = Z_{22} = 0$. The estimation for the gradient of the second distribution moment can be derived using the following expression for x'_g:

$$4x'_g = (1+y)\left[(1+\varepsilon)Z_{22} + (1-\varepsilon)Z_{21}\right] + (1-y)\left[(1+\varepsilon)Z_{12} + (1-\varepsilon)Z_{11}\right]$$

In the above expressions

$$Z_{k_p k} = \sqrt{l_{k_p k}}$$

in order to provide the equality

$$R = \overline{x'^2_g}^{m_n}$$

The minimum criterion for R under condition $p_1 r_1 = p_2 r_2$ is determined in the following way. The gradient R^* estimation (7.17) using adjustable coefficients is determined by (9.6) with A, B and C from (7.18). The estimation of gradient R^* along λ is determined in the form of estimation of the first moment of distribution for the transformed discrete error according to (7.19) and (7.20):

$$\frac{\partial x''_g}{\partial \lambda} = \frac{1}{4}(\varepsilon - y)\left[(-2A_1 + C_1)(\varepsilon - 1) + (2B_1 + C_1)(\varepsilon + 1)\right] + \frac{1}{2}(\varepsilon + y)C_1\varepsilon \tag{9.8a}$$

Expressions (9.6), (7.18), (7.20), and (9.8a) form the basis for the corresponding closed-loop neural network.

The use of Z-transformation and equations (9.8) and (7.21) give the following estimation for gradient R^* along λ:

$$\frac{\partial x''_g}{\partial \lambda} = \frac{1}{4}(\varepsilon + y)\left[Z'_{11}(\varepsilon - 1) + Z'_{12}(\varepsilon + 1)\right] + \frac{1}{4}(\varepsilon - y)\left[Z'_{22}(\varepsilon - 1) + Z'_{21}(\varepsilon + 1)\right] \tag{9.9}$$

where $Z'_{k_p k}$ is determined by (7.22).

The minimum criterion for R under condition $p_1 r_1 = \text{const.}$ is determined in the following way. The gradient R^* estimation (7.24) using adjustable coefficients is determined by (9.6) with A, B and C from (7.25). The estimation of gradient R^* along λ is determined in the form of the estimation of the first moment of distribution for the transformed discrete error according to (9.8a) with A_1, B_1 and C_1 from (7.26). The use of Z_1-transformation and equations (9.8), (7.27), (9.9), and (9.28) give the estimation for gradient R^* along a_i and λ.

9.7 Implementation of Minimum Average Risk Function Criterion for Neurons with Continuum Solutions and K_p Solutions

According to (7.30), in the case of a neuron with a solution continuum (two pattern classes), one obtains

$$x'_g = \frac{1}{2}(1+\varepsilon)Z_{12}\left(x_g\right) + \frac{1}{2}(1-\varepsilon)Z_1\left(x'_g\right) = \frac{1}{2}(1+\varepsilon)\sqrt{l_2(y)} + \frac{1}{2}(1-\varepsilon)\sqrt{l_1(y)} \tag{9.9a}$$

where

$$y = \varepsilon - x_g = F(g) = F\left(\sum_{i=0}^{N} a_i x_i\right)$$

The expression for the estimation of the average risk function gradient through the current neural network signals is obtained after some transformations in the following form:

$$\frac{\partial x'^2_g}{\partial a_i} = \frac{1}{2}\overline{\frac{\mathrm{d}F(g)}{\mathrm{d}g} x_i \left[(1+\varepsilon)\frac{\mathrm{d}l_2(y)}{\mathrm{d}y} + (1-\varepsilon)\frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\right]}^{m_n} \tag{9.10}$$

In this particular case,

$$l_1(y) = (1+y)^2 \quad ; \quad l_2(x_k) = (1-y)^2$$

$$l_1\left(-1-x_g\right) = \left(1-1-x_g\right)^2 = x_g^2$$

$$l_2\left(1-x_g\right) = \left(1-1+x_g\right)^2 = x_g^2$$

$$x'_g = \frac{1}{2}(1+\varepsilon)x_g + \frac{1}{2}(1-\varepsilon)x_g = x_g$$

Consequently,

$$\overline{\frac{\partial x'^2_g}{\partial a_i}}^{m_n} = -2\overline{\frac{\mathrm{d}F(g)}{\mathrm{d}g} x_i x_g}^{m_n}$$

which corresponds to the neuron with α_{2g} minimization analyzed in Sect. 9.2. Expression (9.10) gives the known expression for the estimation of gradient R in the case of two pattern classes and a neuron with two solutions in the form (9.6a).

In the case of continuum pattern classes,

$$x'_g = Z_1\left(x_g\right) = \sqrt{l\left[\left(\varepsilon - x_g\right)\varepsilon\right]}$$

$$\frac{\partial x'^2_g}{\partial a_i} = \overline{\frac{\partial l(y,\varepsilon)}{\mathrm{d}y}\frac{\mathrm{d}F(g)}{\mathrm{d}g}x_i}^{m_n} \tag{9.11}$$

Expression (9.10) for two pattern classes is a particular case of (9.11). The function

$$\frac{\partial l(y,\varepsilon)}{\partial y}$$

in (9.11) must be given a priori.

In the case of a neuron with K_p solutions (K pattern classes), the output signal has the form

$$y = F_\mathrm{p}(g) = 1 + \frac{1}{2}\sum_{k_\mathrm{p}=1}^{K_\mathrm{p}-1}\left[\operatorname{sign}\left(g - a_{k_\mathrm{p}k_\mathrm{p}+1}\right)+1\right]$$

$$g = \sum_{i=0}^{N} a_i x_i$$

Similar to the previous case,

$$\frac{\partial x'^2_g}{\partial a_i} = \frac{\partial}{\partial y}l(y,\varepsilon)\frac{\partial y}{\partial a_i} \tag{9.12}$$

where $l(y,\varepsilon)$ is a $(K\times K)$-matrix with the elements representing the first order difference of the corresponding discrete function $l(x_k,\varepsilon)$. This matrix has the following form in the particular case

$$\left[\frac{\partial l(y,\varepsilon)}{\partial y}\right] = \begin{bmatrix} 0 & 1 & 1 & \dots & 1 \\ -1 & 0 & 1 & \dots & 1 \\ -1 & -1 & 0 & \dots & 1 \\ . & . & . & \dots & . \\ -1 & -1 & -1 & \dots & 0 \end{bmatrix} \tag{9.13}$$

In the expression (9.12),

$$\frac{\partial y}{\partial a_i} = \frac{1}{2}\sum_{k_\mathrm{p}=1}^{K-1}\frac{\partial}{\partial a_i}\operatorname{sign}\left[g - a_{k_p k_{p+1}}\right] = \frac{1}{2}\sum_{k_\mathrm{p}=1}^{K-1}\lim_{B\to\infty}\frac{2}{\pi}\frac{Bx_i}{1+B^2\left(g - a_{k_\mathrm{p}k_\mathrm{p}+1}\right)}$$

Consequently,

$$\operatorname{sign}\frac{\partial y}{\partial a_i} = \operatorname{sign} x_i \quad \text{and finally} \quad \frac{\partial x_g}{\partial a_i} = \frac{\partial l(y,\varepsilon)}{\partial y} = \operatorname{sign} x_i$$

9.8 Implementation of the Minimum Average Risk Function Criterion for Neural Networks with N^* Output Channels (Neuron Layer)

Let us consider closed-loop neural networks with N^* output channels. The optimal models of such neural networks and their secondary optimization functionals were analyzed in Chap. 6 and 7. The case of equal dimensionality of $\boldsymbol{\varepsilon}$ and $\mathbf{x}_k$ is assumed below.

In calculations of discrete error transformations, the output signal has K_0 gradations for each channel. The measured vector of the discrete error has the form

$$(\varepsilon_1,\ldots,\varepsilon_{N^*}) - (y_1,\ldots,y_{N^*}) = (k_1,\ldots,k_{N^*}) - (k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}}) = (x_{1g},\ldots,x_{N^*g})$$

This expression is multiplied by the scalar (7.33), and the norm of the resultant vector is calculated. Then

$$x'_g = \sqrt{x'^2_{1g}+\ldots+x'^2_{N^*g}} = \sqrt{l\left(k_1,\ldots,k_{N^*},k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}}\right)}$$

if

$$(\varepsilon_1,\ldots,\varepsilon_{N^*}) = (k_1,\ldots,k_{N^*})$$

and

$$(y_1,\ldots,y_{N^*}) = (k_{1\mathrm{p}},\ldots,k_{N^*\mathrm{p}})$$

Consideration of the general case of K_0 gradations of the neural network output signal in each channel with the form

$$y_{i^*} = 1+\frac{1}{2}\sum_{k_\mathrm{p}=1}^{K_\mathrm{p}-1}\left[\operatorname{sign}\left(g_{i^*}-a^{i^*}_{k_\mathrm{p}k_{\mathrm{p}+1}}\right)+1\right]$$

$$g_{i^*} = \sum_{i=0}^{N} a_{ii^*}x_i,\quad i^* = 1,\ldots,N^*$$

is not of principle. Let us therefore consider the case $K_0 = 2$:

$$y_{i^*} = \operatorname{sign} g_{i^*}$$

It can be shown that

$$\frac{\partial x'^2_g}{\partial a_{ii^*}} = \frac{\partial}{\partial y_{i^*}} l(\varepsilon_1,\ldots,\varepsilon_{N^*},y_1,\ldots,y_{N^*})\frac{\partial}{\partial a_{ii^*}} y_{i^*} \tag{9.14}$$

Here $l(\varepsilon_1,\ldots,\varepsilon_{N^*},y_1,\ldots,y_{N^*})$ is $(2^{N^*}\times 2^{N^*})$-matrix. The gradient is calculated as the corresponding first order difference along y_{i^*} discrete functions. This matrix can have the form (9.13) in some particular cases. The function $\partial y_{i^*}/\partial a_{ij}$ is determined by its sign

$$\operatorname{sign}\frac{\partial y_{i^*}}{\partial a_{ii^*}} = \operatorname{sign} x_i$$

Let the recognition system have a solution continuum in each of the N^* channels. Let function F be the same for each output channel. Let the transformed discrete error have the first distribution moment equal to the average risk function R. This error is calculated as a sum of squared components of the vector of measured discrete error transformed according to (7.34):

$$x_g'^2 = Z_1\left\{\sum_{i^*=1}^{N^*}\left[\varepsilon_{i^*} - P_{i^*}(\mathbf{x})\right]^2\right\} = l\left[\mathbf{P}(\mathbf{x}),\varepsilon\right]$$

In this case,

$$x_{i^*k} = F\left(g_{i^*}\right) = F\left(\sum_{i=0}^{N} a_{ii^*}x_i\right)$$

and finally

$$\overline{\frac{\partial x_g'^2}{\partial a_{ii^*}}}^{m_n} = \frac{\partial}{\partial x_{i^*k}} l\left(\varepsilon_1,\ldots,\varepsilon_{N^*},x_{1k},\ldots,x_{N^*k}\right)\frac{\partial F\left(g_{i^*}\right)}{\partial g_{i^*}}x_i(n)$$

This expression forms the basis for the design of the corresponding neural network with the closed-cycle adjustment.

9.9 Implementation of the Minimum Average Risk Function Criterion for Multilayer Neural Networks

Three types of closed-cycle adjustment algorithms for multilayer neural networks are represented below as the implementation of the minimum average risk function criterion.

Expression (9.9a) is valid for the neural network of two pattern classes with one output channel ($N^* = 1$) and arbitrary open-loop structure. The estimation of the average risk function gradient has in general the following form:

$$\overline{\frac{\partial x_g'^2}{\partial a_{h_{W-j+1},h_{W-j}}}}^{m_n} = \overline{2x_g'\frac{\partial x_g'}{\partial a_{h_{W-j+1},h_{W-j}}}}^{m_n}$$

Here

$$\frac{\partial x_g'(n)}{\partial a_{h_{W-j+1},h_{W-j}}} = \frac{1}{2}(1+\varepsilon)\left[\frac{1}{2}\frac{1}{\sqrt{l_2(y)}}\frac{\mathrm{d}l_2(y)}{\mathrm{d}y}\frac{\mathrm{d}F(g)}{\mathrm{d}g}\frac{\partial g(n)}{\partial a_{h_{W-j+1},h_{W-j}}}\right]$$
$$+\frac{1}{2}(1-\varepsilon)\left[\frac{1}{2}\frac{1}{\sqrt{l_1(y)}}\frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\frac{\mathrm{d}F(g)}{\mathrm{d}g}\frac{\partial g(n)}{\partial a_{h_{W-j+1},h_{W-j}}}\right]$$

where

$$\frac{\partial g(n)}{\partial \ldots}$$

is determined by the relationship (9.4). Finally

$$\frac{\overline{\partial x_g'^2}^{m_n}}{\partial a_{h_{W-j+1},h_{W-j}}} = \overline{\frac{1}{2}\frac{\mathrm{d}F(g)}{\mathrm{d}g}\frac{\partial g}{\partial a_{h_{W-j+1},h_{W-j}}}\left[(1+\varepsilon)\frac{\mathrm{d}l_2(y)}{\mathrm{d}y}+(1-\varepsilon)\frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\right]}^{m_n}$$

In the particular case of a multilayer neural network with full connections between layers,

$$\frac{\overline{\partial x_g'^2}^{m_n}}{\partial a_{h_{W-j+1},h_{W-j}}} = \overline{\frac{1}{2}\left[(1+\varepsilon)\frac{\mathrm{d}l_2(y)}{\mathrm{d}y}+(1-\varepsilon)\frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\right]} \times \overline{\frac{\mathrm{d}F(g)}{\mathrm{d}g}\sum_{h_{w-1}=1}^{H_{W-1}}\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\nu=1}^{j+1}\left.\frac{\mathrm{d}F(g_{h_{W-\nu}}^{W-\nu})}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}}\right|_{g=g(n)} x_{h_{W-j}}^{W-j}(n)\prod_{\eta=0}^{j+2}a_{h_{W-\eta},h_{W-\eta-1}}}^{m_n}$$

In the case of multilayer neural networks with neurons of two solutions,

$$\frac{\overline{\partial x_g'^2}^{m_n}}{\partial a_{h_{W-j+1},h_{W-j}}} = \frac{1}{2}\left[(1+\varepsilon)\frac{\mathrm{d}l_2(y)}{\mathrm{d}y}+(1-\varepsilon)\frac{\mathrm{d}l_1(y)}{\mathrm{d}y}\right]\operatorname{sign} x_{h_{W-j}}^{W-j} \times\prod_{\eta=0}^{\mathrm{j}+2}\operatorname{sign} a_{h_{W-\eta},h_{W-\eta-1}} \tag{9.15}$$

Consideration of multilayer neural networks with continuum pattern classes and solutions is not of principle. Let us therefore consider the case of K gradations for levels of signals $\varepsilon(n)$ and $y(n)$. The open-loop neural network with $N^* = 1$ is described by the following expression

$$y = 1+\frac{1}{2}\sum_{k_p=1}^{K_p-1}\left[\operatorname{sign}\left(g_{h_W}^W - a_{k_p,k_{p+1}}\right)+1\right]$$

where g_{hW}^W is determined by expression (2.7) in the case of the neural network with a continuum of solutions. The expression for the optimization functional gradient is

$$\frac{\partial x_g'^2}{\partial a_{h_{W-j+1},h_{W-j}}} = \frac{\partial}{\partial y}l(\varepsilon,y)\frac{\partial y}{\partial a_{h_{W-j+1},h_{W-j}}}$$

Matrix $(\partial/\partial y)l(\varepsilon,y)$ is determined here in the same way as it was done in Sect. 9.7. Taking into account that

$$\frac{\partial y}{\partial a_{h_W,h_{W-1}}} = \operatorname{sign} x_{h_{W-1}}^{W-1}$$

the expression for the estimation of the average risk function gradient in the case of the neural network with neurons of two solutions will have the following form:

$$\frac{\partial x_g'^2}{\partial a_{h_{W-j+1},h_{W-j}}} = \frac{\partial}{\partial y} l(\varepsilon,y) \operatorname{sign} \prod_{\eta=1}^{j-1} \sum_{h_{W-1}=1}^{H_{W-0}} a_{h_{W-\eta+1},h_{W-\eta}} x_{h_{W-j}}^{W-j}(n) \tag{9.16}$$

This expression represents a basis for the design of the corresponding closed-loop neural network.

Let us consider the neural network with N^* output channels and two gradations of the output signal by amplitude in each channel. Here

$$y_{i^*} = \operatorname{sign} g_{i^*} = \operatorname{sign} g_{h_W}^W \quad ; \quad h_W = 1,\ldots,N^*$$

Then, having a $(2^{N^*}\times 2^{N^*})$-matrix

$$\frac{\partial}{\partial y_{i^*}} l\left(\varepsilon_1,\ldots,\varepsilon_{N^*},y_1,\ldots,y_{N^*}\right)$$

one can obtain the adjustment algorithm similar to that of Sect. 9.7 for the last layer neuron and Sect. 9.8 for the rest of the neurons.

9.10 Development of Closed-Loop Neural Networks of Non-Stationary Patterns

The main difference as compared with the case of stationary patterns emerges here in the design of the neural network adjustment algorithm. Let us consider a one-dimensional case of the neural network with minimization of α_{2a} in the closed cycle.

In this case,

$$\overline{x_a^2(n\Delta T)}^{m_n} = \overline{\varepsilon^2(n\Delta T)}^{m_n} + \overline{x^2(n\Delta T)}^{m_n} + \overline{a_0^2(n\Delta T)}^{m_n}$$
$$-\overline{2\varepsilon(n\Delta T)x(n\Delta T)}^{m_n} + \overline{2a_0(n\Delta T)\varepsilon(n\Delta T)}^{m_n} - \overline{2a_0(n\Delta T)x(n\Delta T)}^{m_n}$$

The averaging here must be performed across the set of realizations of the non-stationary random process at the time instant $n\Delta T$. Usually one has only a single realization. The value

$$\overline{x_a^2(n\Delta T)}$$

is obtained by averaging across time in the memory interval m_n with an additional constraint of a possible convergence of the process to the stationary state and a priori information about modification of parameters for the distribution of the non-stationary random signal. It must be assumed that the neural network parameters (adjustable coefficient a_0) are constant in the averaging interval m_n in order to express the functional gradient estimation in the algebraic form. In this case,

$$\frac{\mathrm{d}x_a^2(n\Delta T)}{\mathrm{d}a_0} = \overline{2x_a^2(n\Delta T)}^{m_n}$$

The learning algorithm in the non-stationary case is determined by the following relationship:

$$a_0\left[\left(\frac{n}{m}+m\right)\Delta T\right] = a_0\left[\frac{n}{m}\Delta T\right] + K^*\overline{x_a^2(n\Delta T)}^{m_n}$$

In order to design a closed-loop neural network, one needs the information about the character of changes of the signal $x_a(n\Delta T)$ distribution. This information can be unambiguously obtained by the information about the character of changes in the input signal distribution parameters on the interval of the adjustment unit memory as well as information about the neural network structure. If one assumes that the pattern assemblages are normally distributed with a time-dependent mathematical expectation and that the random and determinate components of the random signal $x_a(n\Delta T)$ are statistically independent, then the hypothesis for the mathematical expectation changes is the same as in the case of the $\mathbf{x}(n\Delta T)$ signal. The optimal filtering of the non-stationary signal and the aforementioned hypothesis for the first moment of distribution allows one to use in the adjustment unit the same filter as in the case of estimation of the secondary optimization functional gradient. The synthesis of such filters is considered in [9-7]. Hypotheses about the character of changes of the first moments of distributions in the neural network memory interval are the same for both classes. If it is not so, then the hypothesis of the higher order for the synthesis of the estimation filter of

$$\overline{x_a(n\Delta T)}^{m_n}$$

must be taken.

The analysis of corresponding expressions in the case of non-stationary patterns shows that the estimation of the secondary optimization functional gradient is the problem of the random signal filtering. Characteristics of the non-stationary implementation of the secondary optimization functional were determined above under the assumption of the existence of some a priori information about characteristics of non-stationary input patterns. However, this method is rather complicated in the case of multidimensional and multilayer neural networks and secondary optimization functionals related to the discrete error. To overcome this complexity, we introduce additional a priori information about the neural network input signal. This information allows one to reveal that the class of non-stationary characteristics of pattern

assemblages for which the a priori information about the character of time changes of gradient distribution parameters is sufficient. On the one hand, such an approach simplifies the filter synthesis procedure in the adjustment block. On the other hand, it allows one to design the closed-cycle adjustment algorithms with coefficient correction at each step n. In the previous cases, such a correction was performed after each m_n steps.

The results of multidimensional filter synthesis given in [9-8] can be used for the neural networks with open-cycle adjustment. They can also be used for the design of the neural networks with closed-cycle adjustment at the estimation of secondary optimization functional gradients in the case of the non-stationary pattern neural networks.

9.11 Development of Closed-Cycle Adjustable Neural Networks with Cross and Backward Connections

Let us consider below only the case of the second discrete error distribution moment in the capacity of the secondary optimization functional.

The two-layer open-loop neural network with cross connections for a pattern recognition system is described by the following expression (see Chap. 2):

$$y = F\left[\sum_{j=1}^{H_1} a_j F\left[\sum_{i=0}^{N} a_{ij} x_i\right] + \sum_{i=0}^{N} a_i x_i\right]$$

Similar to the above cases, here

$$\overline{\frac{\partial x_g^2}{\partial a}}^{m_n} = \overline{-2x_g \frac{\partial}{\partial a} y}^{m_n} \tag{9.17}$$

In this case,

$$\frac{\partial y}{\partial a_j} = \frac{\mathrm{d}F(g)}{\mathrm{d}g} y_j \;\; ; \;\; \frac{\partial y}{\partial a_{ij}} = \frac{\mathrm{d}F(g)}{\mathrm{d}g} a_j \frac{\mathrm{d}F(g_j)}{\mathrm{d}g_i} x_i \;\; ; \;\; \frac{\partial y}{\partial a_i} = \frac{\mathrm{d}F(g)}{\mathrm{d}g} x_i$$

These expressions form the basis for the design of the corresponding closed-loop neural network.

The open-loop neural network in the form of neurons with backward connections is described by the following expression (Chap. 2):

$$y(n) = F\left[\sum_{i=0}^{N} a_i x_i(n) + a_k y(n-1)\right] \tag{9.18}$$

Let us consider the case of $m_n = 1$. If $m_n = \text{const.}$, then only the condition of $y(n-1)$ independent of a_i must be satisfied. The Eq. (9.18) gives

$$\frac{\partial y(n)}{\partial a_i} = \frac{\mathrm{d}F(g)}{\mathrm{d}g} x_i(n) \;\; ; \;\; \frac{\partial y(n)}{\partial a_k} = \frac{\mathrm{d}F(g)}{\mathrm{d}g} y(n-1)$$

Consequently, taking into account (9.17), the recurrent relationship for the design of the corresponding closed-loop neural network is

$$\begin{bmatrix}\mathbf{a}(n+1)\\ a_k(n+1)\end{bmatrix}=\begin{bmatrix}\mathbf{a}(n)\\ a_k(n)\end{bmatrix}+\overline{K^{*}\frac{\mathrm{d}F(g)}{\mathrm{d}g}x_g(n)\begin{bmatrix}\mathbf{x}(n)\\ y(n-1)\end{bmatrix}}^{m_n} \tag{9.18a}$$

Let us consider the two-layer neural network with backward connections. The description of the open-loop neural network is the following:

$$\begin{aligned} &y(n)=F[g(n)]\ ;\quad g(n)=\sum_{j=1}^{H_1}a_j y(n)+a_k y(n-1)\\ &y(n)=F[g_j(n)]\ ;\quad g_j(n)=\sum_{i=1}^{N}a_{ij}x_i(n)+a_{kj}y(n-1)+a'_{kj}y(n-1)\end{aligned} \tag{9.18b}$$

Using the transformation (9.17), one gets

$$\begin{aligned} &\frac{\partial y(n)}{\partial a_j}=\frac{\mathrm{d}F(g)}{\mathrm{d}g}y_j(n)\ ;\quad \frac{\partial y(n)}{\partial a_k}=\frac{\mathrm{d}F(g)}{\mathrm{d}g}y(n-1)\frac{\partial y(n)}{\partial a_{ij}}=\frac{\mathrm{d}F(g)}{\mathrm{d}g}a_j\frac{dF\left(g_j\right)}{dg_j}x_i(n)\\ &\frac{\partial y(n)}{\partial a_{kj}}=\frac{\mathrm{d}F(g)}{\mathrm{d}g}a_j\frac{\mathrm{d}F\left(g_l\right)}{\mathrm{d}g_j}y(n-1)\ ;\quad \frac{\partial y(n)}{\partial a'_{kj}}=\frac{\mathrm{d}F(g)}{\mathrm{d}g}a_j\frac{\mathrm{d}F\left(g_j\right)}{\mathrm{d}g_j}y_j(n-1)\end{aligned} \tag{9.18c}$$

These expressions form the basis for the design of the corresponding closed-loop neural network. They can be easily generalized for the case of the neural network with cross and backward connections, of the neural network with arbitrary number of layers, and of the neural network with cross and backward connections of different "logical depths".

9.12 Development of Closed-Loop Neural Networks in the Learning Modes with Arbitrary Teacher Qualification

The learning algorithms similar in quality to the algorithms for the recovery of distribution density are given in [9-5]. These algorithms can be obtained from the given calculation at each step of the adjustment procedure for the multilayer neural network with fixed structure. The adjustment is performed by the vector coordinates of corresponding modes $f(\mathbf{x})$. However, another approach similar to the one used above at the stage of the learning mode is also possible. The average risk in this case is the first moment of signal x'_k distribution (7.35). Consequently,

$$\frac{\partial y'}{\partial a}=\frac{\partial}{\partial a}\rho[\mathbf{x}-\mathbf{b}(y)]=-\frac{\partial \rho}{\partial[\mathbf{x}-\mathbf{b}(y)]}\frac{\partial \mathbf{b}}{\partial y}\frac{\partial y}{\partial a} \tag{9.19}$$

In particular, at $\rho(\mathbf{x},\mathbf{b}) = ||\mathbf{x}-\mathbf{b}||^2$:

$$\frac{\partial \rho}{\partial y}=2[\mathbf{x}-\mathbf{b}(y)]^T\frac{\partial \mathbf{b}(y)}{\partial y}$$

The equation for the unknown functions $\mathbf{b}(y)$ represents some recurrent relationship $(m_n = 1)$:

$$\mathbf{b}(y,n) = \mathbf{b}(y,n-1) + K^* \frac{\partial}{\partial \mathbf{b}} \rho[\mathbf{x} - \mathbf{b}(y,n-1)] \tag{9.20}$$

Equations (9.19) and (9.20) form the basis for the design of the neural networks adjustable by the closed cycle in the self-learning mode. Derivative $\partial y/\partial a$ in (9.19) is determined in the learning mode for the neural network with arbitrary structure. In the case of K_p solutions,

$$\frac{\partial \mathbf{b}(y)}{\partial y}$$

is $(K_p \times N)$-matrix obtained by the solution of (9.20) at threshold current step.

A more detailed development of the closed-loop neural networks with K_p solutions and N^* output channels in the self-learning mode is given in Chap. 12.

The multilayer neural network adjustment algorithm in this case is the following:

1. The initial value $y(0)$ is calculated by the current input signal $\mathbf{x}(0)$ having some initial values of the neural network adjustable coefficients;
2. The column of matrix $b(y,0)$ corresponding to $y(0)$ is selected;
3. The neural network coefficients are adjusted according to (9.19) and so on starting again from point 1.

The $\mathbf{b}(y)$ values at each adjustment step can be determined by parameters and structure of the multilayer neural network. At the arbitrary teacher qualification,

$$x'_g = l(y,\varepsilon)b + \left(1 - b^2\right)\rho[\mathbf{x} - \mathbf{b}(y)] \tag{9.20a}$$

Then

$$\overline{\frac{\partial x'_g}{\partial a}}^{m_n} = \overline{\frac{\partial y}{\partial a}\left\{b\frac{\partial l(y,\varepsilon)}{\partial y} - \left(1 - b^2\right)\frac{\partial \rho}{\partial[\mathbf{x} - \mathbf{b}(y)]}\frac{\partial \mathbf{b}(y)}{\partial y}\right\}}^{m_n} \tag{9.20b}$$

$$\overline{\frac{\partial x'_g}{\partial \mathbf{b}(y)}}^{m_n} = \overline{\frac{\partial}{\partial \mathbf{b}(y)}\rho[\mathbf{x} - \mathbf{b}(y)]}^{m_n} \tag{9.20c}$$

These two equations form the basis for the design of the closed-loop neural network with arbitrary structure and arbitrary teacher qualification. The adjustment algorithm is divided into two independent parts. One of them is determined by the term $\partial y/\partial a$. It depends upon the open-loop neural network structure and determines the quality of the recognition solution.

The developed methods for the multilayer neural network adjustment can also be used for the adjustment of the neural network with several layers and fixed coefficients.

The multilayer neural network adjustment procedure (8.1) provides only the local extremum of the optimization functional. The initial values of the adjustable parameters must be given randomly in the interval determined by some physical arguments. In this case, the full adjustment algorithm of the multilayer neural network must consist of the η^0-volume set of injection stages of random initial conditions, the following stages of adjustment (8.1) and the stage of adjustment results averaging across η^0 (see Chap. 8 and 12).

9.13 Expressions for the Estimations of the Second Order Derivatives of the Secondary Optimization Functional

The expressions for the estimation of the second order derivatives of the second discrete error distribution moment (the secondary optimization functional) are given below for the multilayer neural networks of different types. In the case of a solution continuum,

$$\begin{aligned}
&\overline{\frac{\partial x_g^2}{\partial a_i}}^{m_n} = \overline{-2x_g \frac{\partial F}{\partial g} x_i}^{m_n} \\
&\overline{\frac{\partial x_g^2}{\partial a_i \partial a_j}}^{m_n} = \overline{2x_i x_j \left[\left(\frac{\partial F}{\partial g}\right)^2 - x_g \frac{\partial^2 F}{\partial g^2}\right]}^{m_n}
\end{aligned} \tag{9.21}$$

In the case of a multilayer neural network with sequential connections

$$\begin{aligned}
&\frac{\overline{\partial x_g^2}^{m_n}}{\partial a_{h_{W-j+1},h_{W-1}}} = -2 \sum_{h_{W-1}=1}^{H_{W-1}} \cdots \sum_{h_{W-j+3}}^{H_{W-j+3}} \prod_{\eta=0}^{j+2} a_{h_{W-\eta},h_{W-\eta-1}} \\
&\times \overline{x_g(n) \prod_{\eta=0}^{j+1} \frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}} x_{h_{W-j}}^{W-j}(n)}^{m_n} \frac{\overline{\partial^2 x_g^2}^{m_n}}{\partial a_{h_{w-j+1},h_{W-j}} \partial a_{h_{w-j+1},h_{W-i}}} \\
&= -2 \sum_{h_{W-1}=1}^{H_{W-1}} \cdots \sum_{h_{W-j+3}}^{H_{W-j+3}} \left\{ \frac{\partial}{\partial a_{h_{w-i+1},h_{W-i}}} \left[\prod_{\eta=0}^{j+2} a_{h_{w-\eta},h_{W-\eta-1}} \right] x_g(n) \prod_{\nu=0}^{j+1} \frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}} \right. \\
&\times x_{h_{W-j}}^{W-j}(n) + \prod_{\eta=0}^{j+2} a_{h_{w-\eta},h_{W-\eta-1}} \frac{\partial x_g(n)}{\partial a_{h_{w-j+1},h_{W-i}}} \prod_{\nu=0}^{j+1} \frac{\mathrm{d}F\left[g_{h_{W-l}}^{W-\nu}\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}} x_{h_{W-j}}^{W-j}(n) \\
&+ \prod_{\eta=0}^{j+2} a_{h_{w-\eta},h_{W-\eta-1}} x_g(n) \prod_{l=0}^{j+1} \frac{\partial}{\partial a_{h_{w-i+1},h_{W-i}}} \left[\frac{\mathrm{d}F\left[g_{h_{W-l}}^{W-l}\right]}{\mathrm{d}g_{h_{W-l}}^{W-l}} \right] \prod_{\substack{\nu=0 \\ \nu \neq l}}^{j+1} \frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}(n)} \\
&\left. \times x_{h_{W-j}}^{W-j}(n) + \prod_{\eta=0}^{j+2} a_{h_{w-\eta},h_{W-\eta-1}} x_g(n) \prod_{\nu=0}^{j+1} \frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}} \frac{\partial x_{h_{W-j}}^{W-j}(n)}{\partial a_{h_{w-i+1},h_{W-i}}} \right\}
\end{aligned}$$

Here

$$\frac{\partial\left[\prod\limits_{\eta=0}^{j+2} a_{h_{w-\eta},h_{W-\eta-1}}\right]}{\partial a_{h_{W-i+1},h_{W-i}}}=\begin{cases} 0 \quad \text{if} \quad i>j+3 \\ \prod\limits_{\substack{\eta=0 \\ \eta\neq i+1}}^{j+2} a_{h_{w-\eta},h_{W-\eta-1}} \quad \text{if} \quad i<j+3 \end{cases}$$

$$\frac{\partial x_g(n)}{\partial a_{h_{W-i+1},h_{W-i}}}=\frac{\partial y}{\partial a_{h_{W-i+1},h_{W-i}}}$$

$$=-\sum_{h_{W-1}=1}^{H_{W-1}}\cdots\sum_{h_{W-j+3}=1}^{H_{W-j+3}}\prod_{\eta=0}^{i+2} a_{h_{W-\eta},h_{W-\eta-1}}$$

$$\times\prod_{\nu=0}^{i+1}\frac{\mathrm{d}F\left[g_{h_{W-\nu}}^{W-\nu}(n)\right]}{\mathrm{d}g_{h_{W-\nu}}^{W-\nu}}x_{h_{W-i}}^{W-i}(n)$$

$$\frac{\partial}{\partial a_{h_{W-i+1},h_{W-i}}}\left[\frac{\mathrm{d}F\left[g_{h_{W-l}}^{W-l}\right]}{\mathrm{d}g_{h_{W-l}}^{W-l}}\right]=\frac{\mathrm{d}^2F\left[g_{h_{W-l}}^{W-l}\right]}{\mathrm{d}g_{h_{W-l}}^{2\,W-l}}\frac{\partial g_{h_{W-l}}^{W-l}}{\partial a_{h_{W-i+1},h_{W-i}}}$$

The use of these expressions for the design of corresponding algorithms for multilayer neural networks with a complex structure is complicated. However, the multilayer neural network methods determine the decrease of requirement to estimate the second derivatives of the secondary optimization functional in the case of increasing the complexity of the open-loop neural network structure. For the two-layer system with cross connections,

$$y=F\left[\sum_{j=1}^{H_1}a_jF\left[\sum_{i=0}^{N}a_{ij}x_i\right]+\sum_{i=0}^{N}a_i'x_i\right]$$

$$\overline{\frac{\partial x^2 g}{\partial a}}=\overline{-2x_g\frac{\partial y}{\partial a}}^{m_n}\quad;\quad\overline{\frac{\partial^2 x_g^2}{\partial a_{(1)}\partial a_{(2)}}}=\overline{2\frac{\partial y}{\partial a_{(1)}}\frac{\partial y}{\partial a_{(2)}}-2x_g\frac{\partial^2 y}{\partial a_{(1)}\partial a_{(2)}}}^{m_n}$$

$$\frac{\partial^2 y}{\partial a_i\partial a_j}=\frac{\mathrm{d}^2F(g)}{\mathrm{d}g^2}y_iy_j\quad;\quad\frac{\partial^2 y}{\partial a_j'\partial a_j'}=\frac{\mathrm{d}^2F(g)}{\mathrm{d}g^2}x_ix_j$$

$$\frac{\partial^2 y}{\partial a_i'\partial a_j'}=\frac{d^2F(g)}{dg^2}x_iy_j$$

$$\frac{\partial^2 y}{\partial a_{ij}\partial a_{ml}}=a_j\frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j}x_i\frac{\mathrm{d}^2F(g)}{\mathrm{d}g^2}a_l\frac{\mathrm{d}F(g_l)}{\mathrm{d}g_l}x_m+\frac{\mathrm{d}F(g)}{\mathrm{d}g}a_jx_i\frac{\mathrm{d}^2F(g_j)}{\mathrm{d}g_j}\frac{\mathrm{d}g_j}{\mathrm{d}g_l}x_m$$

$$\frac{\partial^2 y}{\partial a_j \partial a_{ml}} = y_j \frac{\mathrm{d}^2 F(g)}{\mathrm{d}g^2} a_l \frac{\mathrm{d}F(g_l)}{\mathrm{d}g_l} x_m + \frac{\mathrm{d}F(g)}{\mathrm{d}g} \frac{\mathrm{d}F(g_j)}{\mathrm{d}g_j} \frac{\mathrm{d}g_j}{\mathrm{d}g_l} x_m$$

$$\frac{\partial^2 y}{\partial a_i' \partial a_{ml}} = x_i \frac{\mathrm{d}^2 F(g)}{\mathrm{d}g^2} a_l \frac{\mathrm{d}F(g_l)}{\mathrm{d}g_l} x_m$$

In the case of neurons with the feedback loop,

$$\frac{\partial^2 y(n)}{\partial a_i \partial a_j} = \frac{\mathrm{d}^2 F(g)}{\mathrm{d}g^2}\bigg|_{g=g(n)} x_i(n) x_j(n) \quad ; \quad \frac{\partial^2 y(n)}{\partial a_k^2} = \frac{\mathrm{d}^2 F(g)}{\mathrm{d}g^2} \big[y(n-1) \big]^2 \tag{9.22}$$

The obtained expressions form the basis for the design of the multilayer neural network adjustment algorithms with the use of the second derivative of the secondary optimization functional.

Literature

[9-1] Galushkin AI (1973) About adaptation algorithms in multilayer pattern recognition systems. Dokl. AN USSR 1:15–20

[9-2] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energia

[9-3] Viktorov NV, Galushkin AI (1976) Design and investigation of pattern recognition systems under the arbitrary "teacher qualification". Medical radioelectronics, VNII meditsinskoj tekhniki, pp 95–106

[9-4] Galushkin AI, Totchenov VA (1974) "Human-machine" systems in the pattern recognition theory. Proceedings of the seminar "Artificial intelligence. Summary and perspectives", 1974, pp 1–7

[9-5] Tsypkin Ya Z (1968) Adaptation and learning in automatic systems. Moscow, Nauka

[9-6] Tsypkin Ya Z (1970) Foundations of the learning system theory. Moscow, Nauka

[9-7] Galushkin AI, Zotov Yu Ya, Shikunov Yu A (1972) In-line processing of experimental information. Moscow, Energia

[9-8] Galushkin AI (1968) Computation and implementation of optimal filters. Moscow, Automatic control and computer facilities 9:72–128

Chapter 10

Adjustment of Continuum Neural Networks*

This chapter deals with a continuum model of the two-layer neural network with continuum neurons in the first layer and with a neuron with a continuum of features at the input in the second layer. The model structure in this case is described by the expression

$$y(n)=\operatorname{sign}\left[\int_I a_{2l}(i,n)\operatorname{sign}\left(\sum_{l=1}^{L} a_{1l}(i,n)^{*} x_{1l}(n)+a_{10}(i,n)\right)\mathrm{d}i\right] \tag{10.1}$$

Here $x_{1l}(n)$ is the l-th component of the feature vector; y is the neural network output signal; $a_{1l}(i,n)$ is the l-th component of the weighting vector-function of the first layer; and $a_{2l}(i,n)$ is the weighting vector-function of the second layer.

The goal of this chapter is the derivation of the expression for recurrent adjustment procedures of the two-layer continuum neural network weighting coefficients and analysis of peculiarities of these procedures.

The expression for the recurrent procedure for the neuron with a finite number of features is taken as the initial one:

$$\mathbf{a}(n+1)=\mathbf{a}(n)-K^{*}\left.\frac{\partial Y(\mathbf{a})}{\partial \mathbf{a}}\right|_{\mathbf{a}=\mathbf{a}(n)} \tag{10.1a}$$

Here $Y(\mathbf{a})$ is the secondary optimization functional; $\mathbf{a}(n)$ is the system state vector (the current value of the extremum function argument); $K^{*}[L^0\times L^0]$ is the matrix of coefficients; and L^0 is the vector $\mathbf{a}$ dimensionality:

$$\mathbf{a}=\begin{bmatrix} a_1 \\ M \\ a_i \\ M \\ a_n \end{bmatrix} \quad K=\begin{bmatrix} K_{11} & \Lambda & \Lambda & \Lambda & K_{1L^0} \\ M & & & & M \\ K_{i1} & \Lambda & K_{ij} & \Lambda & K_{iL^0} \\ M & & & & M \\ K_{L^0 1} & \Lambda & \Lambda & \Lambda & K_{L^0 L^0} \end{bmatrix}$$

* This chapter is written in collaboration with Fomin Yu. I.

10.1 Adjustment of a Neuron with a Feature Continuum

The relationship (10.1a) can be written in the scalar form

$$a_i(n+1) = a_i(n) - \sum_{j=1}^{L^0} K_{ij} \frac{\partial Y(\mathbf{a})}{\partial a_j}\bigg|_{a_j = a_j(n)} \tag{10.2}$$

The numbers i and j of the vector $\mathbf{a}$ components under $L \to \infty$ are replaced by parameters i and j that are continuously varied in some region I. The summation over j is replaced by an integral that has taken over parameter j. The expression for the recurrent adjustment procedure in the case of the feature continuum is

$$a(i,n+1) = a(i,n) - \int_J K^*(i,j) \frac{\partial Y(a(j))}{\partial a(j)}\bigg|_{a_j = a(j,n)}^{d_j} \tag{10.3}$$

Here $K^*(i,j)$ is the function of two variables i and j.

10.2 Adjustment of the Continuum Neuron Layer

Each neuron from the continuous set in this layer corresponds to some value of parameter i continuously varying in some region J'. One can consider some interval (c, d).

The recurrent adjustment procedure of the continuum neuron layer has the form

$$\mathbf{a}(i,n+1) = \mathbf{a}(i,n) - K^*(i) \frac{\partial Y(\mathbf{a}(i))}{\partial \mathbf{a}(i)}\bigg|_{\mathbf{a}(i) = \mathbf{a}(i,n)} \tag{10.4}$$

Here $K^*(i) - [L^0 \times L^0]$ is the matrix of functions of parameter i, and L^0 is the dimensionality of the feature vector (or vector-function $\mathbf{a}(i)$).

10.3 Selection of the Parameter Matrix for the Learning Procedure of the Continuum Neuron Layer on the Basis of the Random Sample Data

The structure of the neural network continuum layer used as independent system ($a_2(i) = 1$) is described by the expression

$$y = \operatorname{sign} \int \operatorname{sign} \left(\sum_{l=1}^{L_0} a_l(i) x_l + a_0(i) \right) \mathrm{d}i$$

where L^0 is the dimensionality of the feature space.

The recurrent procedure of the neuron layer learning has the following form:

$$\mathbf{a}(i,n+1)=\mathbf{a}(i,n)-K^{*}(i)\frac{\partial Y(\mathbf{a}(i))}{\partial \mathbf{a}(i)}\bigg|_{\mathbf{a}(i)=\mathbf{a}(i,n)} \tag{10.5}$$

We have not yet considered the problem of matrix K^{*} selection. It is evident that $K^{*}(i) = K^{*}(i, n)$ due to its dependence on the adjustment step. Let us analyze the function matrix dependence upon the step number n and parameter i under the condition that $a_2(i) = 1$ for all n (for simplification).

Let us consider the particular case of a diagonal matrix:

$$K^{*}(i)=\begin{Vmatrix} K_{00}(i) & 0 & \Lambda & 0 \\ 0 & K_{11}(i) & \Lambda & 0 \\ \Lambda & \Lambda & \Lambda & \Lambda \\ 0 & \Lambda & \Lambda & K_{LL}(i) \end{Vmatrix}$$

The continuum of neurons realizes a continuum of hyperplanes in the feature space. Each value of parameter i corresponds to the neuron with discriminant function

$$g(y,i)=\sum_{l=1}^{L} a_l(i)y_l + a_0(i) \tag{10.6}$$

In the points of the i-th hyperplane, $g(y,i) = 0$. The distance from some point y to the hyperplane is proportional to $g(y,i)$. Let us term $R(i) = g(y,i)$ as the distance between the point y and the hyperplane corresponding to the parameter i. Let us consider that if y belongs to the first class then the correct orientation for the i-th hyperplane is the orientation providing $R(i) > 0$.

Let at the n-th step of the procedure $R(i) = R(n,i)$, where

$$R(n,i)=\sum_{l=1}^{L} a_l(n,i)y_l + a_0(n,i) \tag{10.7}$$

In the scalar form in this case,

$$a_l(n+1,i)=a_l(n,i)-K_l^{*}\frac{\partial Y(\mathbf{a}(i))}{\partial a_l(i)}\bigg|_{a_l(i)=a_l(i,n)} \tag{10.8}$$

Then

$$R(n+1,i)=\sum_{l=1}^{L} a_l(n+1,i)y_l + a_0(n+1,i) \tag{10.9}$$

can be written in the form

$$R(n+1,i)=\sum_{l=1}^{L}\left[a_l(n,i)-K_l^*(n,i)\frac{\partial Y(\mathbf{a}(i))}{\partial a_l(i)}\bigg|_{a_l(i)=a_l(n,i)}\right]y_l$$
$$+\left[a_0(n,i)-K_0^*(n,i)\frac{\partial Y(\mathbf{a}(i))}{\partial a_0(i)}\bigg|_{a_0(i)=a_0(n,i)}\right]$$

Taking into account (10.9), one obtains

$$R(n+1,i)=R(n,i)$$
$$-\sum_{l=1}^{L}K_l^*(n,i)\frac{\partial Y(\mathbf{a}(i))}{\partial a_l(i)}\bigg|_{a_l(i)=a_l(n,i)}y_l+K_0^*(n,i)\frac{\partial Y(\mathbf{a}(i))}{\partial a_0(i)}\bigg|_{a_0(i)=a_0(n,i)} \tag{10.10}$$

As the sum (integral) of the output signal of the neuron layer is used for the classification of points in the feature space of the neural network, then the point y will be classified correctly in advance at the $(n+1)$-th step if $R(n+1, i)$ is selected in the following way:

$$R(n+1,i)=\begin{cases}-\varepsilon_0 R(n,i)\ , & \text{if} \quad R(n,i)\le 0\\ R(n,i)\ , & \text{if} \quad R(n,i)>0\end{cases} \tag{10.11}$$

(it is assumed that y belongs to the first class). Here $\varepsilon_0>0$. In order to determine $K_l^*(n,i)$, $l=0,\ldots,L$, from (10.10), it is necessary to take the $(L+1)$-th point and to calculate $R(n+1,i)$ for each point according to (10.11). Then the system of $L+1$ linear equations for the function $K_l^*(i)$, $l=0,\ldots,L$, is obtained. Let us rewrite (10.10) in the vector form

$$R(n+1,i)-R(n,i)=-K(n,i)A_{YX} \tag{10.12}$$

Here $K(n,i)=(K_{00}(n,i)\ldots K_{LL}(n,i))$ is the diagonal of matrix $K^*(i)$, and matrix A_{YX} is

$$A_{YX}=\begin{pmatrix}\dfrac{\partial Y(\mathbf{a}(i))}{\partial a_0(i)} & \cdots & \dfrac{\partial Y(\mathbf{a}(i))}{\partial a_0(i)}\\ y_{11}\dfrac{\partial Y(\mathbf{a}(i))}{\partial a_1(i)} & \cdots & y_{1L}\dfrac{\partial Y(\mathbf{a}(i))}{\partial a_1(i)}\\ y_{L1}\dfrac{\partial Y(\mathbf{a}(i))}{\partial a_L(i)} & \cdots & y_{LL}\dfrac{\partial Y(\mathbf{a}(i))}{\partial a_L(i)}\end{pmatrix}$$

It follows from (10.12) that

$$K^*(n,i)=-\big(R(n+1,i)-R(n,i)\big)A_{YX}^{-1} \tag{10.13}$$

This is a final expression for the parameter matrix of functions at the n-th step of the procedure.

One must take m_n samples each consisting of $L+1$ vector for averaging and then select the matrix

$$K^*(i,n)_{res} = \frac{1}{m_n}\sum_{m_n} K^*_{m_n}(i,n) \tag{10.14}$$

The diagonal of matrix $K^*_{m_n}(i,n)$ is calculated by (10.13) for each sample.
The difficulty of this method is the choice of the optimal $\varepsilon_0 > 0$.

10.4 Selection of the Parameter Matrix $K^*(i,j)$ for the Learning Procedure of the Neuron with a Feature Continuum on the Basis of the Random Sample Data

The distance estimation method from the previous section can be also used for the parameter matrix in the case of neurons with a feature continuum. The recurrent learning procedures for the neuron with a feature continuum have the form

$$\begin{cases} a(i,n+1) = a(i,n) - \int_c^{\alpha} K_n(i,j) \left.\frac{\partial Y}{\partial a(j)}\right|_{a(j)=a(j,n)} \mathrm{d}j \\ a_0(n+1) = a_0(n) - K_n \left.\frac{\partial Y}{\partial a_0}\right|_{a_0=a_0(n)} \end{cases} \tag{10.15}$$

The distance from the point $x(n,i)$ to the hyperplane $S(x(i)) = \int a(i)x(i)\mathrm{d}i + a_0 = 0$ is

$$\begin{aligned} R_n(x(n,i)) &= \int a(i)\big|_{a(i)=a(i,n)}\, x(n,i)\mathrm{d}i + a_0(n) \\ R_{n+1} &= \int a(n+1,i)x(n,i)\mathrm{d}i + a_0(n+1) \end{aligned} \tag{10.16}$$

Taking into account (10.15), (10.16), one obtains:

$$\begin{aligned} R_{n+1}(x(n,i)) &= \int_I a(n,i)x(n,i)\mathrm{d}i \\ &\quad - \iint_{IJ} K_n(i,j) \left.\frac{\partial Y}{\partial a(j)}\right|_{a(j)=a(j,n)} \mathrm{d}i\,\mathrm{d}j + a_0(n) - K_n \left.\frac{\partial Y}{\partial a_0}\right|_{a_0(j)=a_0(n)} \\ \Rightarrow R_{n+1}(x(n,i)) &- R_n(x(n,i)) \\ &= \iint_{IJ} K_n(i,j) \left.\frac{\partial Y}{\partial a(j)}\right|_{a(j)=a(j,n)} x(n,i)\mathrm{d}i\,\mathrm{d}j - K_n \left.\frac{\partial Y}{\partial a_0}\right|_{a_0(j)=a_0(n)} \end{aligned} \tag{10.17}$$

If $x(n,i)$ belongs to the first class and $R_n(x(n,i)) < 0$, then one selects $R_{n+1}(x(n,i)) = -\varepsilon_0 R_n(x(n,i))$, $\varepsilon_0 > 0$.

Function $K_n(i,j)$ and coefficient K_n cannot be determined from Eq. (10.17) unambiguously. Let us put some constraints upon function $K_n(i,j)$ introducing the function

$$K_n(i,j)=\frac{1}{K_0^*}\frac{1}{\left.\dfrac{\partial Y}{\partial a(j)}\right|_{a(j)=a(j,n)}}\left.\frac{\partial Y}{\partial a(i)}\right|_{a(i)=a(i,n)} \tag{10.18}$$

where $K_0^* = d–c$.

Equation (10.15) in this case is

$$a(i,n+1)=a(i,n)-\left.\frac{\partial Y}{\partial a(i)}\right|_{a(i)=a(i,n)} \tag{10.19}$$

because taking into account (10.18)

$$\int K_n(i,j)\left.\frac{\partial Y}{\partial a(j)}\right|_{a(j)=a(j,n)}\mathrm{d}i=\left.\frac{\partial Y}{\partial a(i)}\right|_{a(i)=a(i,n)}$$

It follows from (10.19) that function (10.18) is a continuum analogue of the unit parameter matrix K^*. Taking into account (10.18), one obtains for (10.17)

$$R_{n+1}(x(n,i))-R_n(x(n,i))=-\int\left.\frac{\partial Y}{\partial a(j)}\right|_{a(i)=a(i,n)}x(n,i)di-K_n\left.\frac{\partial Y}{\partial a_0}\right|_{a_0=a_0(n)} \tag{10.20}$$

Consequently,

$$K_n=\frac{1}{\left.\dfrac{\partial Y}{\partial a_0}\right|_{a_0=a_0(n)}}\left[R_n(x(n,i))-R_{n+1}(x(n,i))-\int\left.\frac{\partial Y}{\partial a(i)}\right|_{a(i)=a(i,n)}x(n,i)\mathrm{d}i\right]$$

In the case of incorrect classification,

$$K_n=\frac{1}{\left.\dfrac{\partial Y}{\partial a_0}\right|_{a_0(j)=a_0(n)}}\left[(1+\varepsilon_0)R_n(x(n,i))-\int\left.\frac{\partial Y}{\partial a(i)}\right|_{a(i)=a(i,n)}x(n,i)\mathrm{d}i\right] \tag{10.21}$$

One of the possible ways to solve the problem of selection of the parameter coefficients in the case of a neuron with a feature continuum therefore is the solution determined by (10.18) and (10.21). Similar to the previous section, it is possible to use the averaging of K_n and $K_n(i,j)$ across m_n points of the function space.

As in the previous section, the difficulty of this method is the choice of the optimal ε_0.

10.5 Characteristic Properties of the Two-Layer Continuum Neural Network Adjustment Algorithm

These characteristic properties relate to the exchange of the structural matrix of coefficients K^* by the matrix of functions $K^*(i)$ at the transformation to the first layer continuum or by the function of two variables $K^*(i,j)$ at the transformation from the neuron with a finite number of features to the feature continuum in the second layer. The selection algorithm for $K^*(i)$ requires the solution of a system of $L+1$ equations (if $K^*(i)$ is a diagonal matrix of functions). In the case of a finite number of hyperplanes, one needs to find a diagonal matrix of coefficients for each hyperplane, i.e., to solve H systems of $L+1$ equations (H is the number of neurons in the layer).

The algorithm of $K(i, j)$ selection consists in the solution of the integral equation for $K(i, j)$ and coefficient K_n. Figures 10.1 and 10.2 show the open-loop structure of the continuum neural network and the block scheme of its learning algorithm.

10.6 Three Variants of Implementation of the Continuum Neuron Layer Weighting Functions and Corresponding Learning Procedures

This section deals with the open-loop continuum neuron layer structure and its adjustment depending on the method of weighting function implementation.

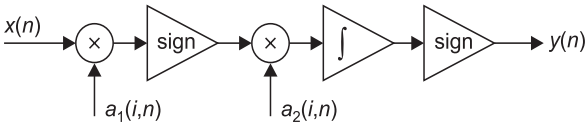


Fig. 10.1.
Open-loop structure of the two-layer continuum neural network

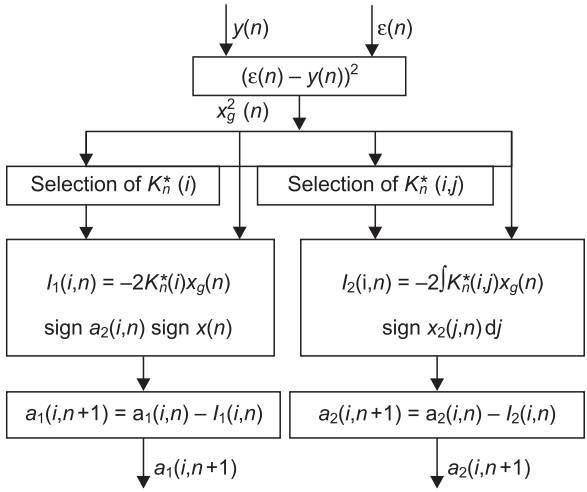


Fig. 10.2.
Block scheme of the two-layer continuum neural network at the $(n+1)$-step

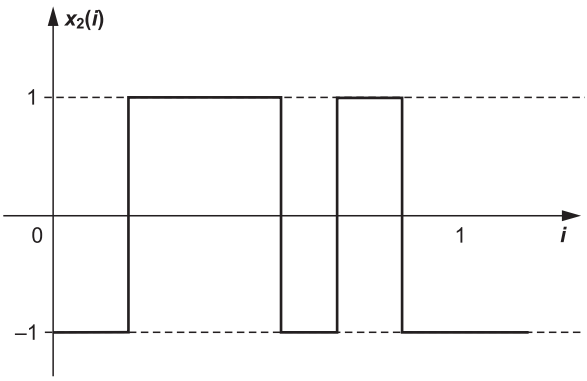


Fig. 10.3.
Output signal of the continuum neuron layer

The open-loop continuum neuron layer structure is described by the expression

$$x_2(i) = \operatorname{sign} \sum_{l=1}^{L} a_l(i)\mathbf{x}_l + a_0(i) \tag{10.22}$$

where $x_2(i)$ is the layer output signal; $\mathbf{x}_l$ is the input signal vector; and $\mathbf{a}(i)$ is the vector of weighting functions for the continuum layer belonging to the piecewise differentiable function class with discontinuity of the first kind and a finite number of zeros ($l = 1 \ldots L$).

It follows from (10.22) that $x_2(i)$ is the function with the form shown in Fig. 10.3. It represents a sequence of rectangles with different durations and unit amplitude. The function under the signum sign

$$f(i) = \sum_{l=1}^{L} a_l(i)\mathbf{x}_l + a_0(i)$$

can have a complex form. However, the form of the function $x_2(i)$ is determined only by the number and location of zeros of the $f(i)$ function. Its form between the neighboring zeros is not important. Hence, the weighting functions $a(i)$ can be approximated rather roughly (Fig. 10.4) or more precisely (Fig. 10.5).

The adjustment procedure is organized in the following way. The approximation interval is divided into S parts. The set of piecewise constant weighting functions (each one determined by S coefficients) or the piecewise linear weighting functions (each one determined by $2S$ coefficients) are constructed thereafter. Then the coefficients that determine the weighting functions are adjusted by means of the gradient method.

The probability of the correct recognition is used as the neural network quality criterion. It is calculated with the use of the test vector sample. If the correct recognition probability under the given number of steps of the learning procedure is less than the required value, then the number S is doubled and the adjustment procedure for the adjustable coefficients is repeated.

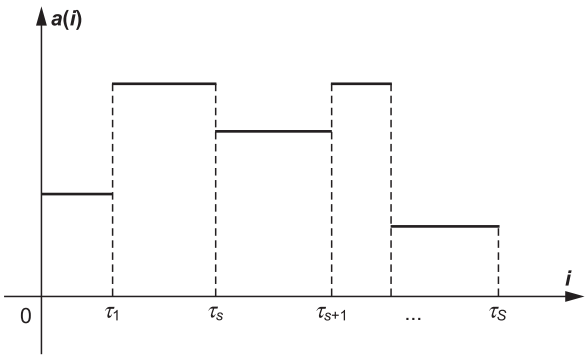


Fig. 10.4.
Piecewise constant approximation of the weighting function of the continuum neuron layer

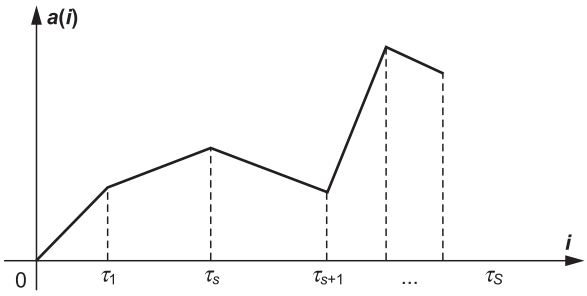


Fig. 10.5.
Piecewise linear approximation of the weighting function of the continuum neuron layer

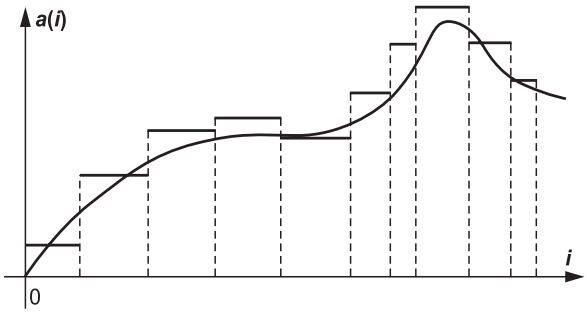


Fig. 10.6.
Weighting function of the continuum neuron layer

As a result, the following important goal is achieved. The layer with the large number of neurons is replaced by the continuum neuron layer, i.e., N weighting coefficients (for each feature) are replaced by the weighting function curve. This weighting function is approximated by the piecewise linear or piecewise constant function required for sufficient recognition probability. The resultant weighting function is described by a sufficiently small number of parameters, at least less than the number of parameters in the case of a discrete neuron layer. The piecewise constant approximation of the weighting function curve is represented in Fig. 10.6.

The maximally monotone function (with minimum number of sign changes of its derivative) provides the minimization of the number of the weighting function approximation intervals using renumbering (ranging) of neurons in the layer.

10.7 Learning Algorithm with α_{2g} Secondary Optimization Functional (the Five-Feature Space) for the Two-Layer Continuum Neural Network

The structure of such a neural network is described by the functional expression

$$y = \operatorname{sign}\left[\int a_2(i)\operatorname{sign}\sum_{l=1}^{5} a_{l1}(i)x_{l1} + a_{10}(i)\,\mathrm{d}i\right] \tag{10.23}$$

Let us consider the case of the functional α_{2g}. The square of the discrete error magnitude is

$$\left|\overline{x_g(i,n)}^{m_n}\right|^2 = \left|\overline{\varepsilon(n)}^{m_n} - \overline{\operatorname{sign} g(n)}^{m_n}\right|^2 \tag{10.24}$$

where

$$g(n) = \int a_2(i,n)\operatorname{sign}\sum_{l=1}^{5} a_{l1}(i,n)x_{l1}(n) + a_{10}(i,n)\,\mathrm{d}i$$

10.7.1 Learning Algorithm for the Second Layer (Feature Continuum Neuron)

Let us calculate the derivative:

$$\frac{\partial\left|\overline{x_g(i,n)}^{m_n}\right|^2}{\partial a_2(i,n)} = -2\overline{x_g(n)}^{m_n}\frac{\partial\left(\overline{\operatorname{sign} g(n)}^{m_n}\right)}{\partial a_2(i,n)}$$

Let us designate

$$\operatorname{sign}\left(\sum_{l=1}^{5} a_{l1}(i,n)x_{l1}(n) + a_{10}(i,n)\right) = x_2(i,n) \tag{10.25}$$

Then

$$\frac{\partial(\operatorname{sign} g(n))}{\partial a_2(i,n)} = \frac{\partial\left(\operatorname{sign}\int a_2(i,n)x_2(i,n)\,\mathrm{d}i\right)}{\partial a_2(i,n)}.$$

The value of the first derivative in this case cannot be determined, and it is necessary to use the information about its sign

$$\operatorname{sign}\left[\frac{\partial(\operatorname{sign} g(n))}{\partial a_2(i,n)}\right] = \operatorname{sign}\left[\frac{2}{\pi}\lim_{B\to\infty}\frac{Bx_2(i,n)}{1+B^2g^2(n)}\right] = \operatorname{sign} x_2(i,n)$$

because

$$\frac{\partial\left(\int a_2(i,n)x_2(i,n)\mathrm{d}i\right)}{\partial a_2(i,n)} = x_2(i,n)$$

(Here and below, the sign of averaging across m_n is omitted in all expressions except the final expressions). Consequently,

$$\frac{\partial\left(x_g(n)\right)^2}{\partial a_2(i,n)} = -2x_g(n)\operatorname{sign} x_2(i,n) \tag{10.26}$$

Taking into account (10.26) and (10.3), one obtains the expression for the recurrent adjustment procedure of the second layer neuron weighting function:

$$a(i,n+1) = a(i,n) + 2\int K(i,j)\overline{x_g(j,n)\operatorname{sign} x_2(j,n)}^{m_n}\mathrm{d}j \tag{10.27}$$

where $x_g(i,n)$ is determined from (10.24), and $x_2(i,n)$ is determined from (10.25).

10.7.2
Learning Algorithm for the First Layer (Continuum Neuron Layer)

Let us calculate the derivative:

$$\frac{\partial\left(x_g(i,n)\right)^2}{\partial a_{l1}(i,n)} = -2x_g(n)\frac{\partial\left(\operatorname{sign} g(i,n)\right)}{\partial a_{l1}(i,n)}$$

$$\frac{\partial\left(\operatorname{sign} g(n)\right)}{\partial a_{l1}(i,n)} = \frac{\partial\left(\operatorname{sign} g(i,n)\right)}{\partial x_2(i,n)} = \frac{\partial\left(x_2(i,n)\right)}{\partial a_{l1}(i,n)}$$

$$\Rightarrow \operatorname{sign}\left[\frac{\partial\left(\operatorname{sign} g(n)\right)}{\partial a_{l1}(i,n)}\right] = \operatorname{sign} a_2(i,n)\operatorname{sign} x_{l1}(n)$$

Consequently,

$$\frac{\partial\left(x_g(n)\right)^2}{\partial a_{l1}(i,n)} = -2x_g(n)\operatorname{sign} a_2(i,n)\operatorname{sign} x_{l1}(n) \tag{10.28}$$

Then one obtains the expression for the recurrent adjustment procedure of the first layer neuron weighting vector-function:

$$\mathbf{a}(i,n+1) = \mathbf{a}(i,n) + 2K^*(i)x_g(n)\operatorname{sign} a_2(i,n)\overline{\operatorname{sign}\mathbf{x}(n)}^{m_n} \tag{10.29}$$

where $K^*(i)$ is the $[L^0\times L^0]$-matrix of functions of parameter i, and L^0 is the dimensionality of the feature vector $\mathbf{x}(n)$.

10.8 Continuum Neuron Layer with Piecewise Constant Weighting Functions

10.8.1 Open-Loop Layer Structure

The interval in which the weighting function is generated is divided into the equal intervals τ. The partition is fixed. The amplitude of rectangles a_s (s is the number of the partition interval) is adjustable in the learning process (Fig. 10.7).

Let us introduce functions

$$h(i)=\begin{cases}1\ , & i>0\\ 0\ , & i<0\end{cases}$$

then $a(i)$ can be represented as

$$a(i)=\sum_{s=1}^{S}a_s\big(h(i-(s-1)t)-h(i-st)\big)$$

Let us introduce the designation

$$H(i,s)=h\big(i-(s-1)t\big)-h(i-st)$$

Then

$$a(i)=\sum_{s=1}^{S}a_sH(i,s) \tag{10.30}$$

The neuron continuum is described by the expression

$$x(i)=\operatorname{sign}\left(\sum_{l=1}^{L}a_l(i)x_l+a_0(i)\right) \tag{10.31}$$

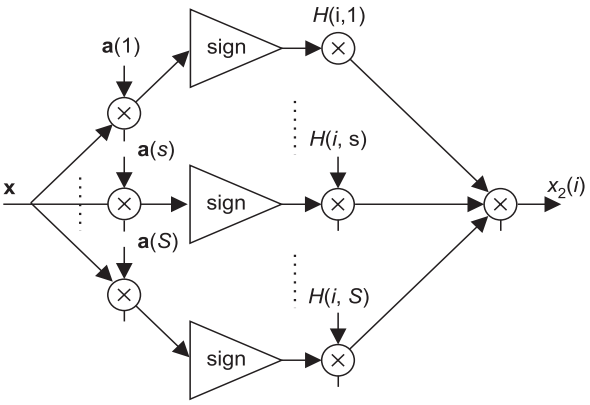


Fig. 10.7.
Diagram of the open-loop structure for the neural network continuum layer with the piecewise constant weighting functions

In (10.31),

$$a_l(i) = \sum_{s=1}^{S} a_{ls} H(i,s)$$

Then

$$x(i) = \operatorname{sign}\left[\sum_{s=1}^{S} H(i,s)\left(\sum_{l=1}^{L} a_{ls}(i)x_l + a_{0s}\right)\right] \tag{10.32}$$

One obtains using the definition of $H(i, s)$ functions

$$x(i) = \sum_{s=1}^{S} H(i,s)\operatorname{sign}\left(\sum_{l=1}^{L} a_{ls}x_l + a_{0s}\right) \tag{10.32a}$$

It follows from (10.32) that this structure represents such a neuron layer

$$x(i) = \sum_{s=1}^{S} H(i,s)\operatorname{sign}\left(\sum_{l=1}^{L} a_{ls}x_l + a_{0s}\right) \tag{10.33}$$

and in this case the neuron outputs connect the input of the next neuron layer in turn during the equal time intervals i. Function $H(i, s)$ is used as a commutation switch of the neuron layer outputs. The structure corresponding to (10.33) is shown in Fig. 10.7.

10.8.2 Recurrent Adjustment Procedure for the Piecewise Constant Weighting Functions

The recurrent adjustment procedure of the continuum neuron layer in the case of a two-layer neural network has the following form:

$$\mathbf{a}(i,n+1) = \mathbf{a}(i,n) + 2K^*(i)\overline{x_g(n)\operatorname{sign} a_2(i,n)\operatorname{sign} x_1(n)}^{m_n} \tag{10.34}$$

Let us consider the particular case of the learning procedure with a diagonal matrix of functions $K^*(i)$.

Let the weighting function of the second layer neuron be piecewise constant. Then at each s-th interval,

$$a_2(i,n) = a_{2s}(n) = \text{const}. \tag{10.34a}$$

since

$$\mathbf{a}_1(i) = \sum_{s=1}^{S} \mathbf{a}_S H(i,s) \tag{10.35}$$

$$K^*(i) = \sum_{s=1}^{S} K_S H(i,s) \tag{10.36}$$

where K_S is a diagonal matrix at the s-th interval. Then the adjustment procedure for the s-th approximation interval will have the form

$$\mathbf{a}_s(n+1)=\mathbf{a}_s(n)+2K_s(i)\overline{x_g(n)\operatorname{sign}a_{2s}(n)\operatorname{sign}x_1(n)}^{m_n} \tag{10.37}$$

10.8.3 About Matrix $K^*(i)$ Estimation

The expression for the diagonal components of matrix $K^*(i)$ was obtained above in the case of the two-layer continuum neural network:

$$K^*(i)=-\left(R_{n+1}(i)-R_n(i)\right)A_{YX}^{-1} \tag{10.38}$$

Here $A_{YX}{}^{-1}= C_{YX}$ sign $a_2(i)$, where C_{YX} is a numerical matrix (independent of i); $a_2(i)$ in this case is a piecewise constant weighting function of the second layer neuron, i.e., it is constant at the s-th interval.

Taking into account (10.35),

$$\begin{cases} R(n)=\sum\limits_{s=1}^{S}R_s(n)H(i,s) \\ R(n+1)=\sum\limits_{s=1}^{S}R_s(n+1)H(i,s) \end{cases} \tag{10.39}$$

Equations (10.36) and (10.39) give for the s-th interval

$$K_s(n)=-\left(R_s(n+1)-R_n(n)\right)A_{YX}^{-1} \tag{10.40}$$

10.9 Continuum Neuron Layer with Piecewise Linear Weighting Functions

10.9.1 Open-Loop Structure of the Neuron Layer

Let us consider the continuum neuron layer with weighting functions shown in Fig. 10.8:

$$a_l(i)=\sum_{s=1}^{S}\left(a_{s,1,l}i+a_{s,0,l}\right)H(i,s) \tag{10.41}$$

The output signal has the form

$$x(i)=\operatorname{sign}\left[\sum_{l=1}^{L}\sum_{s=1}^{S}\left(a_{s,1,l}i+a_{s,0,l}\right)H(i,s)x_l+\sum_{s=1}^{S}\left(a_{s,1,l}i+a_{s,0,l}\right)H(i,s)\right]$$

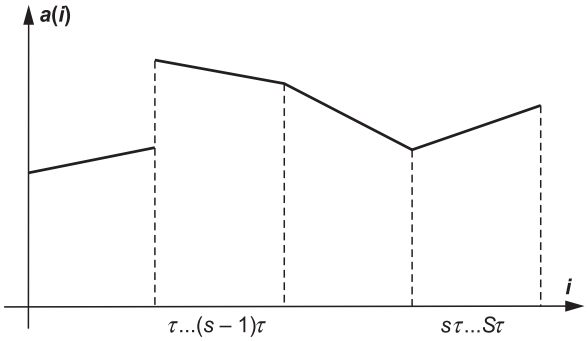


Fig. 10.8.
Weighting functions with piecewise linear approximation for the continuum neural network

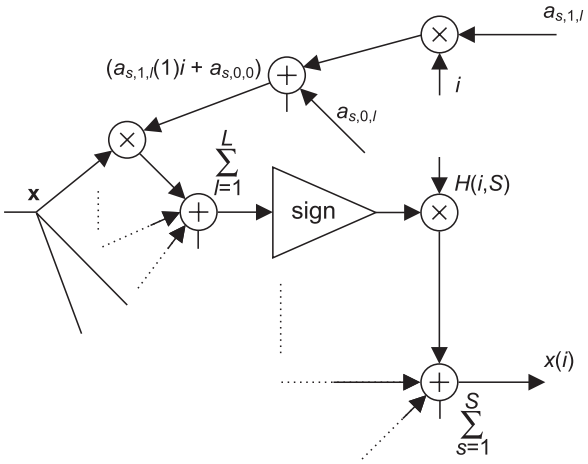


Fig. 10.9.
Diagram of the open-loop structure for the continuum neural network layer with piecewise linear weighting functions (*s*-th channel)

Consequently,

$$x(i)=\sum_{s=1}^{S}H(i,s)\operatorname{sign}\left[\sum_{l=1}^{L}\left(a_{s,1,l}i+a_{s,0,l}\right)x_l+\left(a_{s,1,0}i+a_{s,0,0}\right)\right] \tag{10.42}$$

The structure (10.42) can be considered as the layer of S threshold elements with weighting coefficients linearly dependent on i. The neuron outputs successively connect the input of the next layer by functions $H(i, s)$ at the change of i. The layer structure is shown in Fig. 10.9.

10.9.2
Recurrent Adjustment Procedure for the Piecewise Linear Weighting Functions

Similar to the previous section, matrix $K^*(i)$ has the following form under the condition of the piecewise constant weighting function of the second layer of neurons:

$$K^*(i)=\sum_{s=1}^{S}\left(K_{1s}i+K_{0s}\right)H(i,s) \tag{10.43}$$

Consequently, the layer learning procedure at the s-th interval has the following form:

$$\begin{aligned}\mathbf{a}_{1s}(n+1)&=\mathbf{a}_{1s}(n)+2K_{1s}(n)\overline{x_g(n)\operatorname{sign}a_{2s}(n)\operatorname{sign}x_1(n)}^{m_n}\\ \mathbf{a}_{0s}(n+1)&=\mathbf{a}_{0s}(n)+2K_{0s}(n)\overline{x_g(n)\operatorname{sign}a_{2s}(n)\operatorname{sign}x_1(n)}^{m_n}\end{aligned}\tag{10.44}$$

It is implied here that

$$\mathbf{a}(i)=\sum_{s=1}^{S}\left(\mathbf{a}_{1s}i+\mathbf{a}_{0s}\right)H(i,s)$$

The expression (10.44) is valid only for the aforementioned constraint upon the weighting function of the second layer neuron. In the general case, the learning procedure has the following form:

$$\mathbf{a}_{1s}(n+1)i+\mathbf{a}_{0s}(n+1)i=\mathbf{a}_{1s}(n)i+\mathbf{a}_{0s}(n)i+2K_{1s}^{*}(n,i)\overline{x_g(n)\operatorname{sign}x_1(n)F_s(i,n)}^{m_n}\tag{10.45}$$

where function $F_s(i,n)$ depends on the output signals of the neuron layers following the current one. Consequently, if one were to designate the linear function as

$$f_s(n,i)=2K_s^{*}(n,i)\overline{x_g(n)\operatorname{sign}x_1(n)F_s(i,n)}^{m_n}$$

then one gets the following for the S-th interval:

$$\begin{cases}\mathbf{a}_{1s}(n+1)=\mathbf{a}_{1s}(n)+\dfrac{\mathrm{d}}{\mathrm{d}i}\left(f_s(n,i)\right)\\ \mathbf{a}_{0s}(n+1)=\mathbf{a}_{0s}(n)+f_s(n,i)+\dfrac{\mathrm{d}}{\mathrm{d}i}\left(f_s(n,i)\right)\end{cases}\tag{10.46}$$

The expressions (10.45) and (10.46) are the general adjustment algorithms for the piecewise linear functions of the continuum neuron layer.

The form of matrix (10.43) is derived from the expression (10.38) under the constraints (10.34a) upon the weighting function of the second layer neuron, definition of vector-functions $R_{n+1}(i)$, $R_n(i)$, and condition of linearity for the weighting function at the s-th interval.

In truth, it follows from the definition of vector-functions $R_{n+1}(i)$, $R_n(i)$, and condition of linearity for the weighting function at the s-th interval that

$$\begin{aligned}R_s(n,i)&=R_{1s}(n)i+R_{0s}(n)\\ R_s(n+1,i)&=R_{1s}(n+1)i+R_{0s}(n+1)\end{aligned}\tag{10.47}$$

Expression (10.43) follows from (10.38) and (10.47).

10.10
Continuum Neural Network Layer with Piecewise Constant Weighting Functions (the Case of Fixed "Footsteps")

10.10.1
Open-Loop Layer Structure

Let the weighting coefficients have the form shown in Fig. 10.4 and divisional points τ_s not be fixed. Let the amplitudes of rectangles be fixed at the value $U_{sl}\Delta a$, where U_{sl} is the integer number; Δa is the fixed value. The l-th weighting function of the layer has the following form:

$$a_l(i) = \sum_{s=1}^{S} U_{sl}\Delta a H(i,s) \tag{10.48}$$

where

$$H(i,s) = h(i-\tau_{s-1}) - h(i-\tau_s)$$

The output signal of the layer is

$$x(i) = \operatorname{sign}\left[\Delta a \sum_{l=1}^{L}\sum_{s=1}^{S} U_{sl}H(i,s)x_l + \sum_{s=1}^{S} U_{s0}H(i,s)\right]$$

Consequently,

$$x(i) = \sum_{s=1}^{S} H(i,s)\operatorname{sign}\left[\Delta a\left(\sum_{l=1}^{L} U_{sl}x_l + U_{s0}\right)\right] \tag{10.49}$$

The structure (10.49) is similar to the structure (10.32a) under the condition that time intervals τ_s, during which corresponding neurons connect the input of the next layer, variate, and the amplitudes of weights at the s-th interval are fixed. Thus one obtains an s-dimensional vector of variable parameters τ_s.

10.10.2
Recurrent Adjustment Procedure for Piecewise Constant Weighting Functions with Variable Interval Lengths τ_s.

The recurrent adjustment procedure for vector $\boldsymbol{\tau}$ has the following form:

$$\boldsymbol{\tau}(n+1) = \boldsymbol{\tau}(n) - K^* \left.\frac{\partial Y(n)}{\partial \boldsymbol{\tau}}\right|_{\boldsymbol{\tau}=\boldsymbol{\tau}(n)} \tag{10.50}$$

where $Y(n)$ is the secondary optimization functional. If

$$Y(n) = \left(\varepsilon(n) - y(n)\right)^2 = x_g^2$$

where $y(n)$ is the output neural network signal and $\varepsilon(n)$ is the supervisor instruction, then

$$y(n) = F[x(i)]$$

where $x(i)$ is the output signal of the current layer; F is the transformation performed by the following neural network layers over $x(i)$:

$$\frac{\partial Y}{\partial \tau_s} = -2x_g \frac{\partial y}{\partial x(i)} \frac{\partial x(i)}{\partial \tau_s}$$

Taking into account the form of $x(i)$:

$$\frac{\partial Y}{\partial \tau_s} = -2x_g \frac{\partial F}{\partial x(i)} \frac{\partial x(i)}{\partial H(i,s)} \frac{\partial H(i,s)}{\partial \tau_s}$$

$$\frac{\partial x(i)}{\partial H(i,s)} = \operatorname{sign}\left[\Delta a\left(\sum_{l=1}^{L} U_{sl}x + U_{s)}\right)\right] \tag{10.51}$$

It follows from the function $H(i,s)$ definition:

$$\frac{\partial H(i,s)}{\partial \tau_s} = \frac{\partial}{\partial \tau_s}\left[h\left(t-\tau_{s-1}\right) - h\left(t-\tau_s\right)\right] \Rightarrow \frac{\partial H(i,s)}{\partial \tau_s} = -\frac{\partial h\left(t-\tau_s\right)}{\partial \tau_s}$$

$$\Rightarrow \frac{\partial H(i,s)}{\partial \tau_s} = -\frac{\partial\left(\operatorname{sign}\left(t-\tau_s\right)\right)}{\partial\left(t-\tau_s\right)} \frac{\partial\left(t-\tau_s\right)}{\partial \tau_s}$$

$$\frac{\partial\left(t-\tau_s\right)}{\partial \tau_s} = -1$$

$$\frac{\partial H(i,s)}{\partial \tau_s} = \delta\left(i-\tau_s\right)$$

It is necessary to use the information about the sign of the first derivative

$$\frac{\partial H(i,s)}{\partial \tau_s}$$

$$\operatorname{sign}\left(\frac{\partial H(i,s)}{\partial \tau_s}\right) = 1 \quad , \quad \text{if} \quad i \geq \tau_s$$

Finally,

$$\frac{\partial Y}{\partial \tau_s} = -2x_g \frac{\partial F}{\partial x(i)} \operatorname{sign}\left[\Delta a\left(\sum_{l=1}^{L} U_{sl}x + U_{s0)}\right)\right] \tag{10.52}$$

Literature

[10-1] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energia
[10-2] Galushkin AI (1977) Continuum models of multilayer pattern recognition systems. Automatic and computer technology 2
[10-3] Galushkin AI (1992) Continuum neural networks. Neurocomputer 2:9–14

Chapter 11

Selection of Initial Conditions During Neural Network Adjustment – Typical Neural Network Input Signals

11.1 About Selection Methods for Initial Conditions

The homogeneous neural network adjusted to the solution of some specific problem represents a dynamic system described by difference or differential (linear or nonlinear) equations. Therefore, the problem of selecting initial conditions for the system adjustment is an important part of the neural network theory. The quality of such selection significantly influences the quality of the solution. Usually this aspect is not considered.

It seems that only Rosenblatt [11-1] mentioned this problem. However, he took the zero initial values of coefficients in all the experiments. This does not guarantee the achievement of extremum with a satisfactory value of optimization functional. This is especially the case in the problems with multi-extremum optimization functionals.

One can consider two methods of selection of such initial conditions: selection of random initial conditions and selection of deterministic initial conditions. In the first method, the multi-extremum secondary optimization functional is used. The random elements are introduced into the procedure of the secondary optimization functional extremum search for the search of the local and global extrema of this functional. The local extremum search is necessary for the solution of the problem of the structure minimization for the multilayer neural network. The impression of a too large number of the local extrema in the space of the adjustable coefficient emerges at the first stage of the use of the random initial conditions. However, the enlargement of the open-loop neural network structure results in the increase of the numerosity of the multilayer neural network states. This numerosity is estimated by the secondary optimization functional value. This means that the majority of the local extrema provide one and the same quality of recognition. This remark must be taken into account in the methods described below of the multilayer neural network quality estimation by estimation of the secondary optimization functional value. The experimental results obtained in this chapter and the aforementioned comment show the validity of the approach to the problem of adjustment with the use of the random initial conditions in spite of the fact that such an approach introduces redundancy in the time of the neural network adjustment.

The goal of the second method with deterministic initial conditions is the a priori introduction of the neural network into the region of one of the local extrema of the secondary optimization functional. The multilayer neural network must be maximally amorphous at the level of the first, second, etc., layers, i.e., it must be ready to solve the

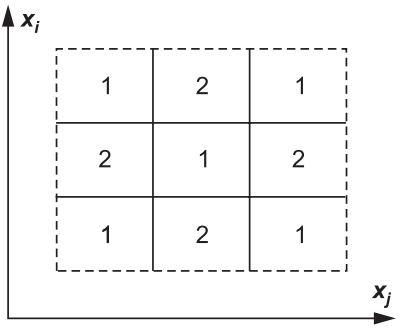


Fig. 11.1.
Divisional surface at the selection of initial conditions: *1* – the first class; *2* – the second class

most complex (from the point of modality of $f(\mathbf{x})$) recognition problem. The preliminary variant of the possible structure of the divisional surface in this case at the problem of learning for recognition of two pattern classes is represented in Fig. 11.1. The final variant can be determined only after introducing the criterion of amorphism or dispersion.

It is evident that the multilayer neural network with the lowest amorphism and dispersion is the neural network with equal coefficients of the first layer neurons and with corresponding divisional surfaces shifted to the "margin" of the feature space. The physically implemented feature space region is shown in Fig. 11.1 by the dotted line. This also takes place in the self-learning mode if there is no preliminary information about the cells in Fig. 11.1 belonging to this or that class. The initial condition for the adjustable coefficients of the second, etc., layers is calculated by the geometry of the divisional surface realized by the neurons of the first layer with instruction concerning belonging of the initial feature space regions to this or that class.

The initial conditions are selected according to the a priori information about distributions at the already-known structure of the open-loop neural network and selected optimization functional. The selection of initial conditions depends not only on the selection of vector $\mathbf{a}(0)$ but also on the method of calculation of parameter matrix K elements at each procedure step. Methods for the selection of initial conditions can be ordered by the form of the used information:

- Random initial conditions without use of the learning sample;
- Deterministic initial conditions without use of the learning sample;
- initial conditions with use of the learning sample.

11.2 Algorithm of Deterministic Selection of the Initial Conditions in the Adjustment Algorithms for Multilayer Neural Networks

This algorithm uses a priori information about mode configuration in the feature space, their number and variance.

Let the feature space be normalized into the unit hypercube

$$K = \{\mathbf{x} : 0 \le x_i \le 1,\ \ i = 1,\ldots,N\}\ ,\quad \mathbf{x}_j \in K\ ,\quad (j = 1,\ldots,M) \tag{11.1}$$

where $\mathbf{x}_j$ is the feature space of j-th pattern of the learning sample; N is the space dimensionality; and M is the learning sample length. Let us consider the case of two classes with K_1 and K_2 modes. Designations of corresponding classes are

$$\mathbf{r}_{1i}(i=1,\ldots,K_1),\ \mathbf{r}_{2i}(i=1,\ldots,K_2)$$

Let us organize monotone sequences for the mode projections on each coordinate axis:

$$0 \le r^j_{s_1 i_1} \le r^j_{s_2 i_2} \le \ldots \le 1 \quad , \quad \left(s_{1,2}=1,2;\ \ j=1,\ldots,N\right), \tag{11.2}$$

where $i_{1,2}$ are the numbers of modes of the first and second classes.

Let us consider differences of the following form:

$$\left| r^j_{1i_1} - r^j_{2i_2} \right| \quad , \quad j=1,\ldots,N \tag{11.3}$$

Let $\sigma_{1,2}$ be the estimation of the variance of j-th mode and let the following condition be valid:

$$\left| r^j_{1i_1} - r^j_{2i_2} \right| < \left(\sigma^j_{1i_1} + \sigma^j_{2i_2} \right) \quad , \quad j=1,\ldots,N \tag{11.4}$$

Then hyperplane is drawn through the middle of the segment

$$\left[r^j_{1i_1},\ r^j_{2i_2} \right] \tag{11.5}$$

square with j-th coordinate axis. If the drawn hyperplane separates also projections of other modes, then it is drawn through the point obtained by the averaging of middles of corresponding segments (11.5) whose edges are separated by this hyperplane. If the condition (11.4) is not fulfilled for a given pair of modes, then the hyperplane is not drawn.

Let us consider an example shown in Fig. 11.2 for illustration. The mode configuration has the following form:

$$0 < r^1_{21} < r^1_{11} < r^1_{12} < r^1_{22} < 1$$

$$0 < r^2_{11} < r^2_{22} < r^2_{12} < r^2_{21} < 1$$

One checks the validity of conditions (11.4) for x_1. As a result, one draws hyperplane *1* through the middle of the segment R^1_1, R^1_2, where

$$R^1_1 = \frac{r^1_{12} - r^1_{21}}{2} \quad , \quad R^1_2 = \frac{r^1_{22} - r^1_{21}}{2}$$

Similarly checking for x_2, one draws the hyperplane 2 through the middle of the segment R^2_1, R^2_2, where

$$R^2_1 = \frac{r^2_{22} - r^2_{11}}{2} \quad , \quad R^2_2 = \frac{r^2_{12} - r^2_{22}}{2}$$

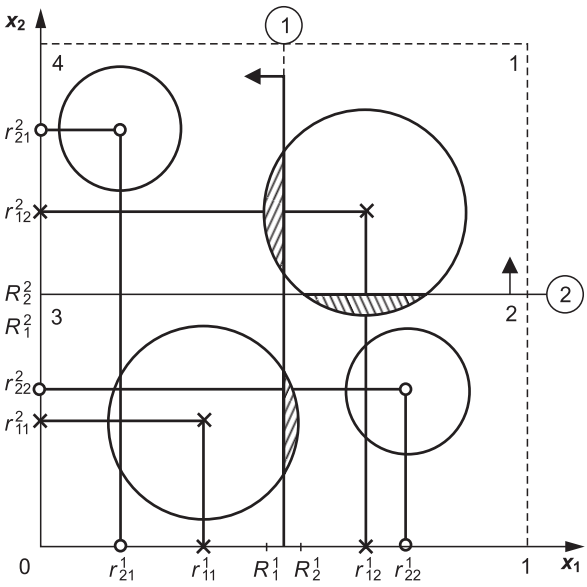


Fig. 11.2.
Example of the use of algorithm for deterministic selection of the initial conditions

Table 11.1. Parameter description

R1	Matrix for estimation of coordinates of first class mode centers
R2	Matrix for estimation of coordinates of second class mode centers (projections of mode centers on *i*-th coordinate axis are located in the *i*-th column)
SGM1	Matrix for estimation of first class mode variances
SGM2	Matrix for estimation of second class mode variances
M1	Number of first class modes
M2	Number of second class modes
N	Dimensionality of feature space
R3	Matrix obtained by combination of matrices R1 and R2 by columns (M3 = M1 + M2)
X	Matrix of points on coordinate axes through which hyperplanes are drawn
K	Number of points on current coordinate axis
I	Number of coordinate axes
A	Matrix of drawn hyperplane coefficients
IK	Number of hyperplanes

Then the first and the third sections can be considered as belonging to the first class, and the second and the fourth sections can be considered as belonging to the second class. The error is determined by the hatched regions.

The block scheme of the program that realizes this algorithm and parameter description is represented in Fig. 11.3 and in Table 11.1.

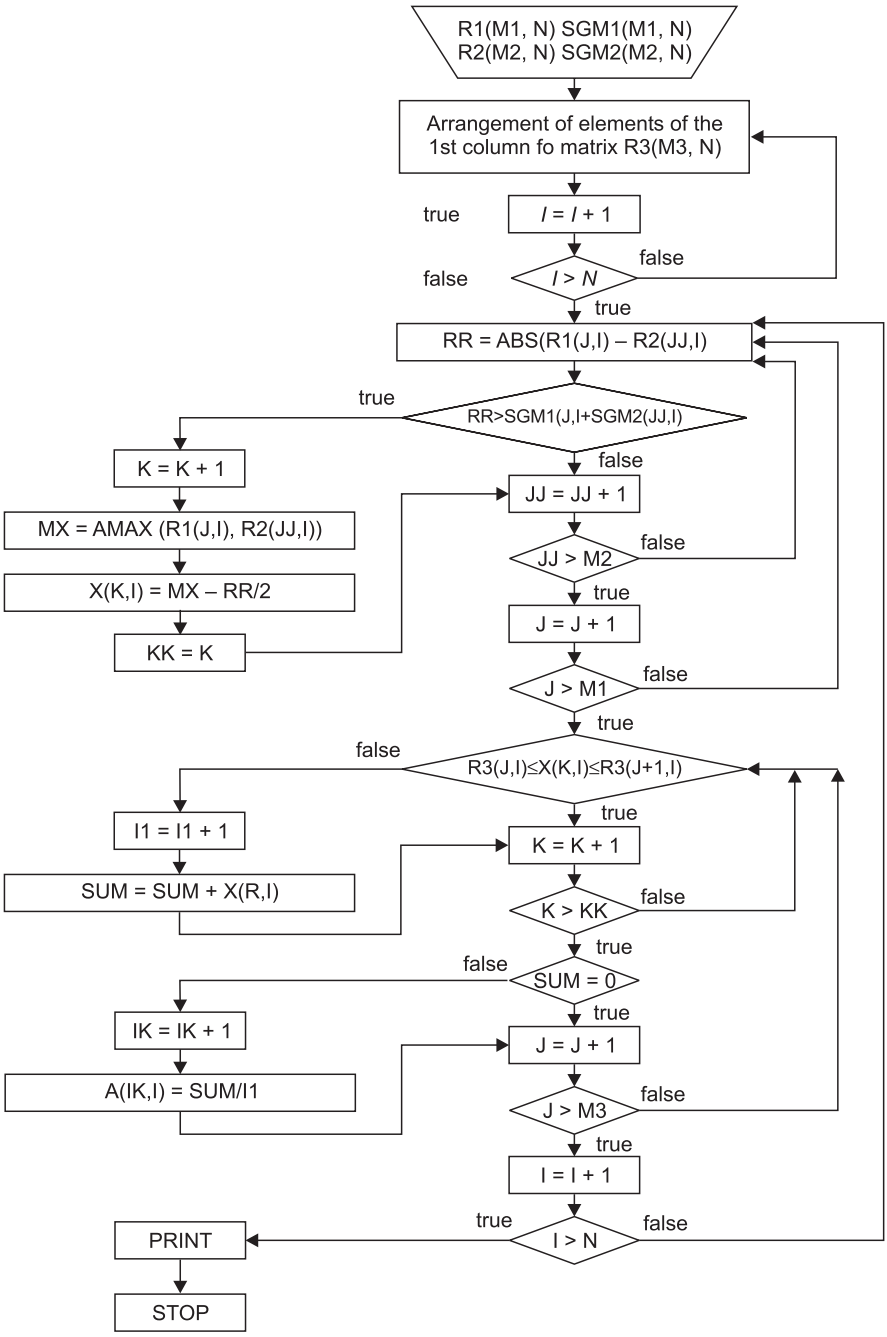


Fig. 11.3. Block scheme of the program realizing the algorithm of deterministic selection of the initial conditions

11.3 Selection of Initial Conditions in Multilayer Neural Networks

The problem of the initial condition selection in multilayer neural networks is divided into a step-by-step selection of the initial conditions for the first, second, etc., layers. The first layer was considered above. For example, let the coefficients of the first layer be determined as a result of deterministic selection, and thus the set of sections is obtained. Each section corresponds to the number consisting of + 1 and –1. If this set provides the location of the divisional surface close to the optimal one (the recognition error is small), then the probability that this section configuration will be changed is small. The estimations of such a probability are given below. The learning samples for the second, etc., layers will be preserved with this probability. The obtained coefficients can remain constant if they correctly perform decomposition of the output space of the previous layer.

Figure 11.4 shows the example of the second and third neuron layer coefficient values dependent upon the classes' number distribution across sections when the configuration is not changed. In the cases *a*–*d*, the logic function is realized on the neuron of the second layer. In the case *e*, the three-layer neural network is realized.

Let us consider the problem of coefficient fixation in more detail. Let some piecewise linear surface be realized in the feature space. Each section must correspond to some class number in order to form the learning sample.

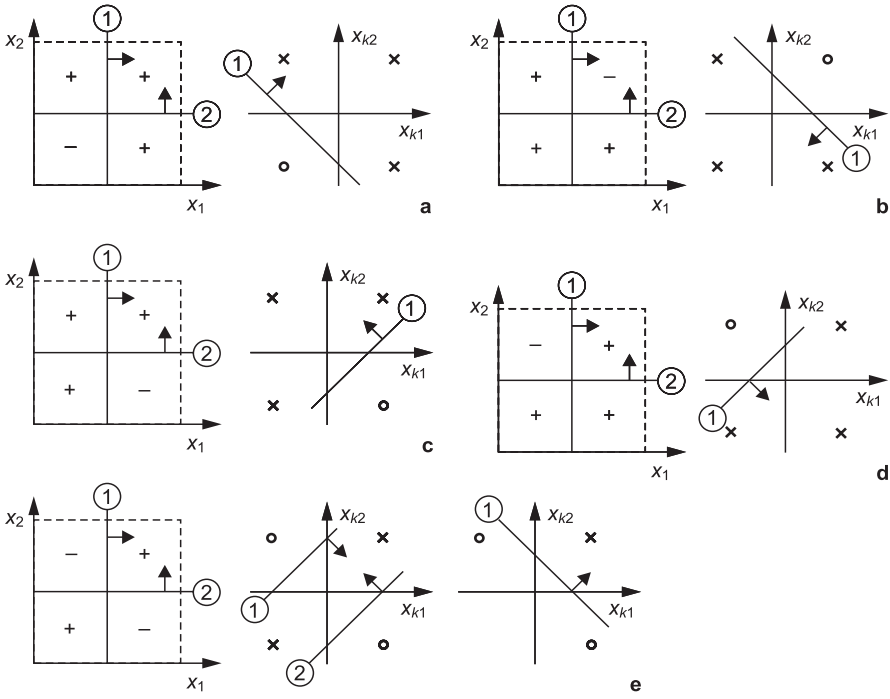


Fig. 11.4. Example of the second and third neuron layer coefficients value dependent upon the classes' number distribution across sections

It can be done in the following way. First, one obtains the correspondence between each element of the learning sample and the section number. Then one determines the class number corresponding to the maximum number of its patterns in each section. And this class number will correspond to this section. Then it is necessary to select optimal coefficients for the second layer. Since the piecewise linear surface in the majority of diagnostic problems is not very complicated, then one can obtain experimental tables of correspondence for the second layer of neurons to the given configuration. Though this task is rather complex and time-consuming, its solution provides the reduction of the initial condition selection to the deterministic selection of the initial conditions for the first layer and fixation of the coefficients of the following layers using such tables. The estimation for the probability of preservation of configuration in the feature space is given below.

Let us consider the deterministic selection of the initial conditions for the multilayer neural network with three neurons in the first layer shown in Fig. 11.5. Table 11.2 represents the values of the logic function Y with the first layer outputs as its arguments. The values of this function on sections are represented for the given configuration. Here y_i ($i = 1,2,3$) are the outputs of the first layer neurons. Such a function can be realized by two neurons of the second layer with coefficients:

$$\begin{aligned} \mathbf{a}_1^2 &= \{1,\ 1,\ 1,\ -2\} \\ \mathbf{a}_2^2 &= \{-1,\ -1,\ -1,\ 2\} \end{aligned} \tag{11.6}$$

Sections in Table 11.2 can be modified in the process of further learning. Their number is $2^{H_1} - m$, where H_1 is the number of neurons in the first layer and m is the number of sections selected as initial conditions. In our example, two sections, $[+1, +1, -1]$ and $[-1, +1, -1]$, can arrear. One obtains under the fixed coefficients of

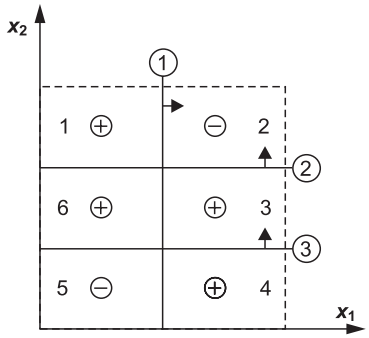


Fig. 11.5.
Example of the deterministic selection of the initial conditions for the multilayer neural network with three neurons in the first layer

Table 11.2.
Values of the logic function Y with the first layer outputs as its arguments

No.	1	2	3	4	5	6
y_1	−1	+1	+1	+1	−1	−1
y_2	+1	+1	−1	−1	−1	−1
y_3	+1	+1	+1	−1	−1	+1
y	+1	−1	+1	+1	−1	+1

the first layer that the new sections belong to the first class. It is seen from Fig. 11.4 that the first section increases the recognition error because the element of the second class will appear there, and the second section does not increase the error. The classification error can increase (as a result of fixation the coefficients of all the layers are higher than the first one) in the following situations: first, if the new sections are considered as belonging to the other pattern class, and second, if the old section that will change its configuration in the learning process gets more other class patterns than before.

Let us consider the section $\varepsilon^i = \{\varepsilon_1^i, \ldots, \varepsilon_k^i\}, \varepsilon_j^i = 1, \ldots, m$. Let the first class correspond to the number of the section ε^i and let this section get S_i elements of the second class in the process of the initial condition selection. Then the probability of the wrong classification obtained due to the surface of this section is S_i/M_i, where M_i is the number of patterns in the section ε^i. Further closed-cycle learning must involve those hyperplanes whose change will decrease the error probability S_i/M_i. The probability of the section deformation in the learning procedure can be defined as

$$p_i = p(S_i/M_i) \tag{11.7}$$

Let us assume that dependence of the section deformation upon the probability S_i/M_i is linear

$$p_i \approx \frac{S_i}{M_i} \tag{11.8}$$

and that patterns are uniformly distributed along the section with the density S_ρ^i. Then,

$$p_i \approx \frac{S_\rho^i V_i}{M_i} \tag{11.9}$$

Evidently, $p_i \to 0$ if $V_i \to 0$, $S_\rho^i = \text{const}$, and $p_i \to 0$ if $S_\rho^i \to 0$, $V_i = \text{const}$. This means that the probability of the section deformation that does not contribute to the recognition error is zero, and that under the constant distribution density S_ρ^i, the decrease of the section volume results in the decrease of the number of elements S_i that it gets thus decreasing the probability (11.9). In the case of $V_i \to V_0$, one gets $p_i \to ½$ (under the assumption that the sample includes for $M/2$ elements of each of two classes). The averaging of (11.9) for all the m sections gives

$$p_m = \frac{1}{m}\sum_{i=1}^{m}\frac{S_i}{M_i}$$

and taking into account that

$$S_i \le \frac{M_i}{2}$$

for each p_i, one gets the estimation for the average section deformation probability under the considered configuration:

$$p_m \leq \frac{1}{2}$$

As a result, it is seen that it is possible to estimate the probability of the already existent section deformation in the learning process at the stage of initial condition selection for the first layer neurons, i.e., to estimate the validity of the use of fixed coefficients of the following layers.

Taking into account the above considerations, one can present the initial condition selection method for the multilayer neural networks.

1. The piecewise linear surface is drawn in the feature space with the help of the deterministic algorithm of the initial condition selection for the first layer neurons;
2. The correspondence between the i-th pattern of the learning sample and the section number ($\varepsilon^j = 1, \ldots, m$) is found with the help of the examination of the first layer across all of the learning sample;
3. Teacher instruction E_i ($i = 1, \ldots, K$) is assigned to the section number ε^j, where K is the number of patterns. The values Y_{ij} ($i = 1, \ldots, p$) are calculated for this purpose for each section (Y_{ij} is the number of patterns of the i-th class that occurred in the j-th section; p is the number of classes whose patterns occurred in the j-th section) and the maximum value max Y_{ij} is found. The corresponding E_i is the required teacher instruction;
4. The probability of considering the j-th section as belonging to the i-th pattern class is calculated:

$$P_j = 1 - \frac{S_j}{M_j}$$

where

$$S_j = M_j - \max_i Y_{ij} \quad , \quad M_j = \sum_{i=1}^{P} Y_{ij}$$

5. The logic function realization on one neuron of the second layer is checked. If this function is realized then the initial condition selection is terminated at the stage of the second layer neuron learning;
6. If this does not take place, then the initial condition selection is performed on the neurons of the second layer either in a way similar to p. 1 or with the help of tables of correspondence of the second layer coefficients to the specified section configurations;
7. A similar procedure is used for the following layers;
8. The sub-samples for the first layer learning are formed: one checks successively for each section ε^j which hyperplanes that form this section contribute to the error S_j, then it shifts each hyperplane by $\pm r_j$ ($r_j > r_j^0$), where r_j^0 is the mean distance between the nearest patterns that occurred in the given section, and it checks the change of S_j. The hyperplanes selected in such a way must be exposed to further closed-cycle learning.

The partial learning of the first-layer neurons provides the following: first, not to train hyperplanes with optimal location, second, to get the reduced learning time.

The aforementioned method reduces the multilayer neural network learning to the learning of the first neuron layer.

11.4 Initial Condition Formation for Neural Network Coefficient Setting in Different Problems of Optimization

Generally, multilayer neural networks in solving mathematical problems form the multiextremum optimization functional in which the connection is the best, and often the only decision is reaching during the setting process using a system of a global extremum of the given functional. In the overwhelming majority of works, the initial condition selection is suggested to be made through the following methods:

- To choose *ground* initial conditions;
- To choose initial conditions of setting casually;

In this work, it is suggested that the initial conditions of setting the multilayer neural network weight coefficients is chosen for solving mathematical problems in a specific way of solving every problem. In the first turn it creates the following problems:

- Systems of algebraic equalities;
- Systems of algebraic inequalities;
- Approximation and extrapolation of functions;
- Pattern recognition, as a particular problem of function approximation;
- Clusterization or self-teaching;
- Optimization;
- Dynamic object modeling.

In the succeeding period, a quantity of current problems in the neural network logical basis will be increased [11-5, 11-6] and correspondingly for other problems the methods of initial condition formation will be developed for a multilayer neural network setting.

The problem of initial condition choice (formation) in its turn is separated into two parts:

- Formation of the main idea (algorithm) of the choice of initial conditions;
- Weight coefficients of the neural network calculation of a chosen structure as initial for setting in an adaptive regime.

Some problems of the initial condition choice in a given work are defined only without a presentation of a final decision; in our opinion this is what is also important for a correct orientation of the researchers in this field.

Here the main goal is defining the problem of initial condition formation for the coefficient setting in the multilayer neural networks by using a specific method for every concrete problem.

11.4.1 Linear Equality Systems

The main idea of the choice of initial conditions for two variants of investigated neural network algorithms to solve the algebraic equalities is presented in Figs. 11.6a and 11.6b.

In the first version of the neural network algorithm of solving the algebraic equalities systems in the case of a one-layer neural network the initial conditions of X vector is offered to calculate, solving N equalities of one variable, forming their coefficients by separation (main) matrix A diagonal, or N equalities of two variables, forming their coefficients by separation of two matrix A diagonals (main and next to it from above or below). One can consider the version of three matrix A diagonals separately (main and two next). The choice of the diagonal number in the matrix is determined by admissible time for solving the diagonal system equalities using ordinary methods for $X(0)$ calculating the following solution of linear equality systems by a neural network algorithm. As mentioned above numerous times, this procedure is effective for big dimension systems. In this case, the problem is the initial condition choice of weight coefficients in the second, third etc. layers of the multilayer neural network, which solves the system of algebraic equalities in the structure presented on Fig. 11.6a.

During the work, [11-6] a modified approach to neural network algorithms of the solution of algebraic equality systems is offered, which is presented in Fig. 11.6b. Here the neural network input is A matrix and b vector, and output – required X solution. In conclusion, the main problem is, in forming the output signal of neural networks that is found by the first method of algebraic equality system determination, the calculatation of the neural network coefficients, which could content to ratio $X(0) = F\{W(0)\}$, where F is the conversion made by the neural network, and $W(0)$ is the number of initial values of weight coefficients of the neural network by a defined structure.

11.4.2 Linear Inequality Systems

By solving linear inequality systems, the initial condition choice of setting multilayer neural network coefficients is possible to make. This is also possible with a linear equality system solution with proper signs of inequality substitution in the inequality systems to the signs of equalities.

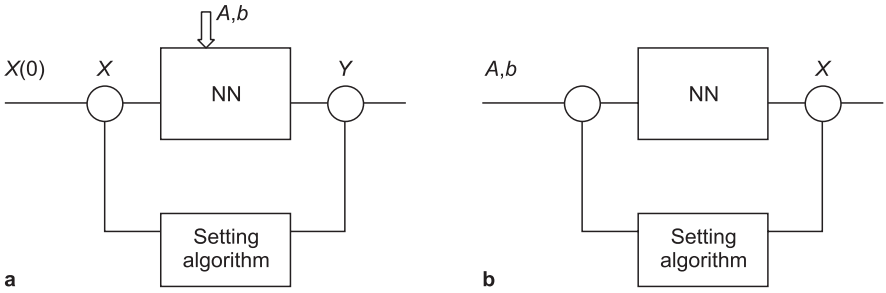


Fig. 11.6. Initial conditions choice for two variants of investigated neural network algorithms

11.4.3
Approximation and Extrapolation of Functions.

It is widely believed that neural networks are mostly effective approximators and extrapolators of functions. It is directly connected with the natural development of the theory of filtration and extrapolation signals from linear theory, where the amount of a priori information about useful signals and noises is quite considerable in nonlinear theory. In this connection a neural network machine using the filtration and the signal extrapolation problem includes and summarizes quite a great quantity of attempts to solve the nonlinear processing problem by means of other methods. Neural networks solve the filtration and signal extrapolation problems for difficult and unknown characteristics of a useful signal, often variables in time, and variables in time-noise characteristics. In the framework of neural networks are the initial attempts to build multivariable filters and extrapolators.

From our point of view, the natural choice for initial conditions of a multilayer neural network setting of equivalent linear filter in adaptive approximators and function extrapolators is the adaptive filter construction. Linear discrete filter Zadeh/Ragozini is a *z*-filter of an order which is equal to the memory of a filtration or extrapolation system with coefficients that calculate as functions of N memory and extrapolation time $\acute{a}$. Because the linear filter is produced based on a priori information about known functional form signal and additive noise to a useful signal, particularly in the form of white noise, a few sets of weight coefficient filters can exist for useful signals of different complexity and correspondingly calculate with different computational complexity. The main problem is to calculate the *z*-filter coefficients to the appropriate initial coefficients of a neural network of a fixed structure.

11.4.4
Pattern Recognition

At the outset is the problem of choosing initial conditions for an adaptive neural network setting; solving the problem of pattern recognition appeared in the work 11.2]. In this work two methods of choice of mentioned initial conditions are considered: the choice of accidental initial conditions and the choice of determinate initial conditions. The choice of Accidental initial conditions is made because of the secondary optimization functional multi-extreme coupled with the multi-modality of distribution $f(x)$ input signal and limitation of the open-ended neural network structure. Accidental elements are introduced into the procedure of the extremum search of the secondary optimization functional because of the necessity of local and global extremum of the mentioned functional search. The necessity of the local extremum search is determined by the necessity to solve the problem of the multilayer neural network structure minimization by analysis of setting results. At the first phase of accidental initial condition use (and the following phase of setting equalization results for a multitude of accidental overshoot phases of initial conditions) an impression is made about a great number of local extrema of the secondary optimization functional in space of adjustable coefficients. However because it is necessary to mark that with meshing of an open-ended neural network structure, the multiplicity of multilayer neural network conditions increases, which is estimated by the value of the secondary optimization functional. In other words, the majority of local extremum

functionals in the space of adjustable coefficients ensure the same recognition quality. This remark is necessary to connect with the methods described below of a multilayer neural network quality estimate by the secondary optimization functional value estimated by the current signals in the neural network. Taking into account everything that has been mentioned above, one can mark the accuracy of the approach to the setting using the accidental initial conditions, although this approach obviously introduces the redundancy in the neural network setting time with the intention of input signal full learning (particularly, solving the global extremum functional).

The purpose of determinate initial condition leading is a priori neural network leading into the area of one of the local extrema of the secondary optimization functional in the space of adjustable coefficients. On the geometry level, the first, second etc. multilayer neural network should be maximumly amorphed, distributed, or namely, advanced to solve the most difficult (from the view of $f(x)$) modality of the recognition problem. Thinkable configuration of a shared surface in this case with teaching of recognition of two class patterns is presented in Fig. 11.7, although it is the preliminary version. The final version can be determined only by leading of an amorphous criterion, distribution. Obviously that which is smallest amorphed and distributed is a multilayer neural network where all the coefficients of the first layer of neurons are similar, and appropriately discriminate surfaces are removed to the edge of the feature space. This is physically realized in the neural network area of the feature space in Fig. 11.7 and is indicated with a dotted line. This also spreads to the self-teaching regime, if the cell belonging in the Fig. 11.7 to one or another class was not pointed out before. Initial conditions for adjustable coefficients of the second layer etc. are estimated by the geometry of the dividing surface, which is realized by neurons of the first layer with specification of the initial feature space areas belonging to one or another class. One of the methods of choosing determinate initial conditions was offered in work [11-2].

In conclusion of this point it is necessary to mark those methods which are used in the initial condition choice that one can sort by the type of information utilization:

- Accidental initial condition choice without the use of teaching extracts;
- Determinate initial condition choice without the use of teaching extracts.

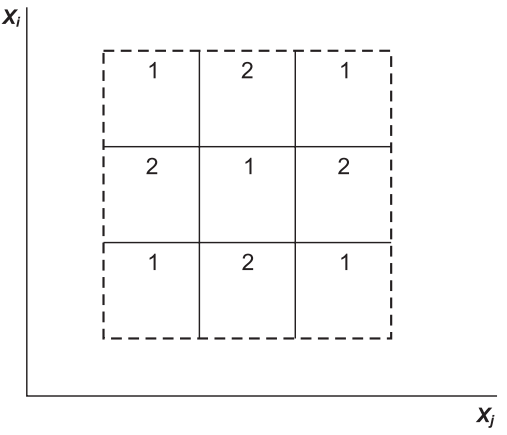


Fig. 11.7.
Dividing surface with choice of initial conditions:
1 – first class; 2 – second class

The means of choosing initial conditions with the use of extracts is connected with the possibility of using the initial coefficient neural network utilization of the neural network with fixed structure of results of teaching neural network with a variable structure [11-2, 11-4].

11.4.5 Clusterization

The first problem is choosing initial conditions for an adaptive neural network setting which resolves the problem of clusterization that was applied in work [11-2]. The natural version of the initial condition choice in the adaptation process in the multilayer neural networks, solving the problem of clusterization, is the construction of the initial dividing surface in the multidimensional feature space as it was shown in Fig. 11.7 without indicating which areas belong to one or another class. Weight coefficients of the first and following neurons are formed based on the geometry of the assumed dividing surface with equal dissection of the multidimensional feature space to the areas which are appropriate to the estimated clusters.

11.4.6 Traveling Salesman Problem

The traveling salesman problem is the particular problem of linear programming at that time when the linear programming problem is the particular problem of optimization together with quadratic and non-linear programming. The traveling salesman problem is solved by finding the shortest route between N towns when leaving the appointed town and returning back there.

A possible version of the choice of initial conditions, of the initial route while solving the traveling salesman problem is the choice of route, which is logically choosing from the appointed town the nearest one to it, then the nearest to the chosen one, etc.

11.4.7 Dynamic System Modelling

Dynamic system identification using the neural networks is mostly effective in the case of substantially nonlinear systems, systems with variable parameters and structure, and the multivariate and classified systems. In the easiest case of linear systems, the identification is realized by feeding the system of jump signals of every amplitude to the input and solving z-transformation of the transition process. In that case, the model of the system is a z-filter with coefficients of a z-transition function. In the case of more complicated systems, the initial information specified above of the object one can get, gives to the input the sequence of jump signals of different amplitudes in the range of input signal changes from zero to X_{Bxmax} (in the Fig. 11.8). At the same time, in consequence of the material nonlinear object, the reactions to the different jump signals won't be linear-contiguous. At the same time, one can use z-transformation of system reactions to the jump input signals of different amplitudes for forming the initial values of coefficients of the first layer of a multilayer neural network, which identifies the ex-

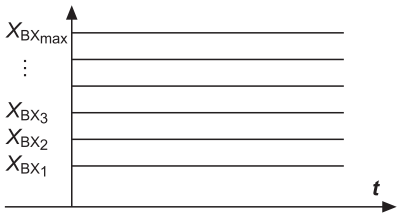


Fig. 11.8.
Choosing initial conditions of adaptation in the multilayer neural networks by solving the problem of dynamic systems identification

plored object. The subject matter is still the question of forming the initial values of coefficients of the consequent layer of the multilayer neural network with full sequential connections, and also the feed-back connection coefficients in the case of utilization of neural network identification for networks with adjustable feedback connections.

11.4.8 Conclusion

The choice of initial conditions for a multilayer neural network setting is an important means of both speeding up the convergence of adaptive algorithms and ensuring the convergence to the global extremum optimization functional. Unfortunately, practically no attention is paid in this matter of the classic computational mathematics to the development of iterative algorithms of complex problem solving. Every current problem in the neural network logical basis demands its method of forming initial conditions for the adaptive algorithm of the multilayer neural network setting. It will be the matter of developing this research in the future by solving a class of problems, which are solved on the neural network logical basis.

11.5 Typical Input Signal of Multilayer Neural Networks

The selection of some class of typical signals is performed for the objective comparison of the multilayer neural network quality in the adjustment mode and in the stationary state. This problem is solved in a relatively complete form in the case of the linear systems of automatic control with deterministic and random input signals. The relatively complete class of deterministic input signals is the class of polynomial input signals usually used for the estimation of the control system's quality. The main characteristics of the signal complexity here is the corresponding polynomial exponential order or the distribution $f_x(\mathbf{x})$ modality. It is reasonable to consider that in the case of self-learning, the distribution of the typical stationary input signal of the multilayer neural network is multimodal with a relatively homogeneous location of distribution $f_x(\mathbf{x})$ modes in the physically realized pattern space.

Figure 11.9 shows the complete class of the typical multilayer neural network input signals in the self-learning mode illustrated by the isolines of $f_x(\mathbf{x})$ in the physically realized pattern space (the two-dimensional representation in the X space is conditional). Here r is the complexity of the typical input signal of the multilayer neural network. The variance of each $f_x(\mathbf{x})$ mode must be chosen in such a way that the modes are sufficiently pronounced. Figure 11.10 shows isolines $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ for the typical

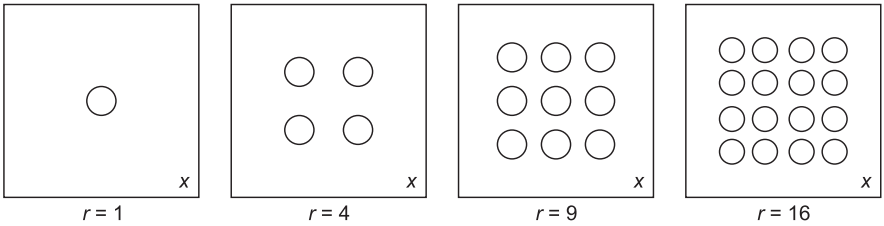


Fig. 11.9. Conditional representation of the typical neural network input signals in the self-learning mode ordered by the degree of complexity

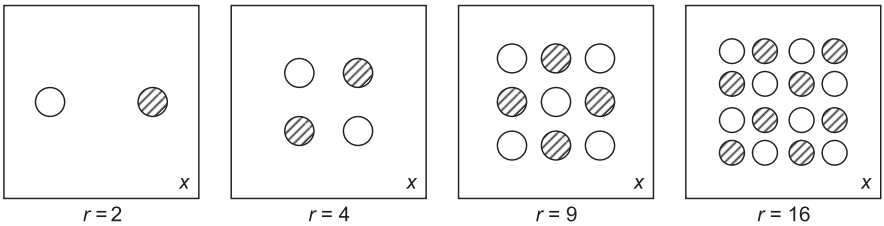


Fig. 11.10. Conditional representation of two classes of the neural network input signals in the learning mode ordered by the degree of complexity

input signals in the case of multilayer neural network learning of the recognition of two pattern classes (f_1 – empty circles, f_2 – shaded circles).

It must be mentioned that each specific problem solved by the neural network requires its own method of the choice of typical input signals.

Literature

[11-1] Rosenblatt F (1964) Principles of neurodynamics. Moscow, Mir
[11-2] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energia
[11-3] Galushkin AI, Sudarikov VA, Shabanov EV (n.y.) Mathematic modelling
[11-4] Galushkin AI (2000) Neural networks theory. (series "Neurocomputers and their application", book 1), M., IPRZHR, 2000
[11-5] (in co-authorship) (2002) Neuromathematics, series "Neurocomputers and their application", book 6, M., IPRZHR, 2002
[11-6] Galushkin AI (2003) Neuromathematics (problems of development), Neurocomputer 1
[11-7] Galushkin AI (1968) Calculation and design of optimal discrete filters. articles. Automatic control and computing machinery 9, Machgiz
[11-8] Galushkin AI (1968) Neurocomputers and neuromathematics. Washington
[11-9] Butenko AA, Galushkin AI, Pantyukhin DV (2002) Neural network methods of linear algebraic equations system solving. VIII All-Russian conference «Neurocomputers and their application», NKP-2002, M., 21–22 March 2002
[11-10] Pantyukhin DV (2002) Neural network methods of linear algebraic equations system solving. XLV Scientific conference of Moscow Institute of Physics and Technology (State University) "Modern problems of fundamental and applied sciences". Moscow-Dolgoprudniy, 29–30 November 2002, part 1, p. 79
[11-11] Pantyukhin DV (2003) Version of neural network algorithm of linear algebraic equations system solving. 5 International conference "Digital signals processing and its application" DSPA-2003, 12–14 March 2003, Moscow, Russia, vol. 2, p. 607

Chapter 12

Analysis of Closed-Loop Multilayer Neural Networks

12.1 Problem Statement for the Synthesis of the Multilayer Neural Networks Adjusted in the Closed Cycle

This chapter represents the final stage for the synthesis of multilayer neural networks with fixed structure that are adjusted in the closed cycle. It is implied that the open-loop neural network structure, the general characteristics of the signal, and the multilayer neural network adjustment algorithm are given. Several problems must be solved for the quality estimation of the closed-loop multilayer neural networks.

The first one is the selection of initial conditions for the adjustment of the multilayer neural network weighting coefficients. Two methods of the initial condition selection are considered: random selection with the averaging of the results across the number of random injections and search of all local and global extrema, and a deterministic method with placing the neural network into the region of a global extremum of the secondary optimization functional by means of defining some piecewise linear divisional surface at the initial stage.

The second problem is the selection of the typical input signal class for the multilayer neural network for the estimation of their functioning quality in the transient and stationary modes. The complexity of the input signal will be particularly determined by the modality of the conditional distribution $f'(\mathbf{x}/\varepsilon)$.

The third problem is the selection of the parameter matrix K^* in the algorithm of the extremum search for the secondary optimization functional. This problem can be solved analytically and by means of statistical modeling methods. The general analytical methods for the closed-loop neural networks consist of the following steps:

1. Determination of the density of probability distribution for the estimation of the secondary optimization functional gradient vector;
2. Derivation of the stochastic differential equation for the change of the distribution density of the adjustable neural network coefficients in the adjustment process;
3. Solution of this equation;
4. Search of the primary optimization functional distribution parameters by means of integrating over the feature space and over the neural network state space.

With the result of the analysis according to the aforementioned steps and according to the requirement to provide the given quality estimated by the primary functional value, one can solve the problem of synthesis of the neural network adjustment circuit. Notice that the analytical solution of the third step is rather complicated. That is why such methods are illustrated in the present study only by some particular examples. The statistical analysis is considered to be the main one.

12.2 Investigation of the Neuron Under the Multi-Modal Distribution of the Input Signal

12.2.1 One-Dimensional Case – Search Adjustment Algorithm

The neuron with two solutions and with minimization of α_{2g} was modeled. The block diagram of the modeled system is shown in Fig. 12.1. The possibility of the design of closed-loop systems with the search adaptation procedure was analyzed. The assemblages of the first and second class patterns have multi-modal distributions. This is the case of the system structure insufficiency. In such a case, the structure complexity is less than the complexity of the solved problem, and therefore the potential recognition quality cannot be achieved in principle.

Figure 12.2 represents distribution densities of the first and second class assemblages and the dependence of the average density function α_{2g} upon the threshold a_0 in the case when the left neuron indicates the first class region, and the right neuron indicates the second class region. The gradient α_{2g} in the search procedure was calculated according to the following expression:

$$\frac{\mathrm{d}\widehat{\alpha}_{2g}}{\mathrm{d}a_0} = \frac{\alpha_{2g}(a_0+\Delta a_0)-\alpha_{2g}(a_0-\Delta a_0)}{2\Delta a_0}$$

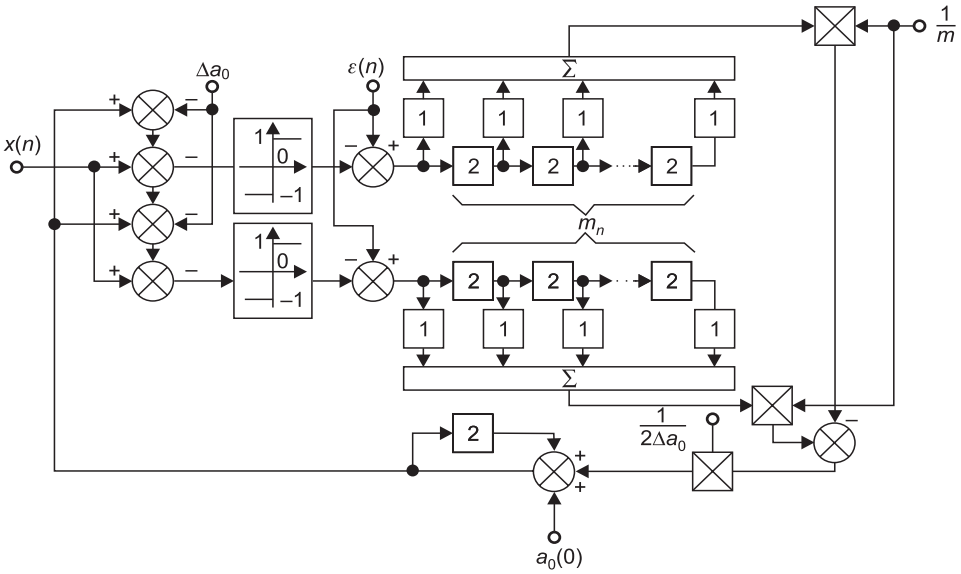


Fig. 12.1. Block scheme of the search neural network adjustable in the closed cycle with minimization of the discrete error second moment: *1* – square-law function generator; *2* – *T*-cycle delay unit

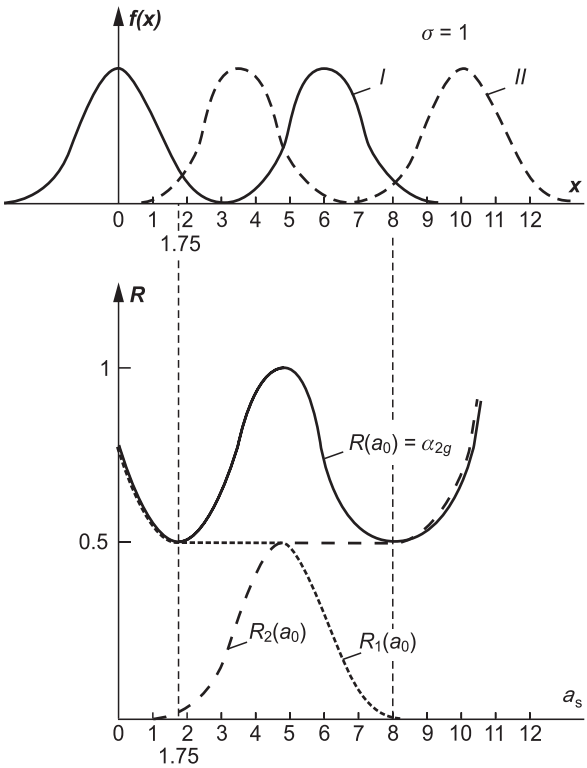


Fig. 12.2.
The input signal and optimization functional characteristics: *I* – the first class; *II* – the second class

where Δa_0 is the amplitude of the search oscillations. The estimation of $d\alpha_{2g}/da_0$ was performed by means of averaging across m_n realizations of the system input signal. The main aim of the modeling is the estimation of the influence of Δa_0, K^*, m_n, $a_0(0)$ upon the dynamics of the adjustment circuit of the coefficient a_0. The results of modeling are the following:

1. The search oscillations are suitable for the design of the neural network closed-cycle adjustment block. The higher value of Δa_0 results in the higher precision of the adjustment circuit performance in the stationary state (Fig. 12.3);
2. The higher value of K^* results in the lower value of the systematic error for the iteration procedure of the optimal solution search and in the higher value of the random error of this procedure (Fig. 12.4);
3. The higher value of m_n results in the lower value of the random errors and in the higher values of dynamic errors in the adjustment circuit (Fig. 12.5);
4. With any initial conditions $a_0(0)$ (Fig. 12.6), the iteration procedure of the optimal solution search converges to one of the local extrema. The results of the algorithm performance with the introduction of the random elements into the search procedure are represented in Fig. 12.7.

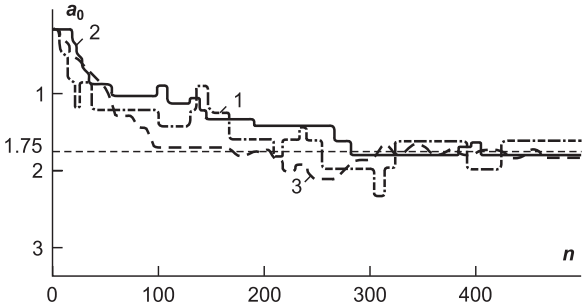


Fig. 12.3.
The investigation of the step value influence upon the system adjustment dynamics when $K = 0.5$; $m_n = 20$; $a_0(0) = 0$: *1* – $\Delta a_0 = 0.25$; *2* – $\Delta a_0 = 0.5$; *3* – $\Delta a_0 = 1$

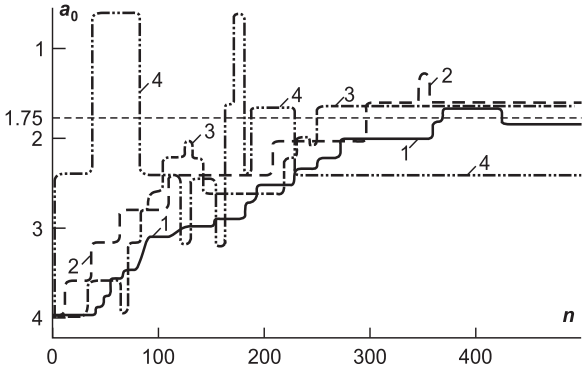


Fig. 12.4.
The investigation of the K^* value influence upon the system adjustment dynamics when $\Delta a_0 = 0.25$; $m_n = 20$; $a_0(0) = 4$: *1* – $K^* = 0.25$; *2* – $K^* = 0.5$; *3* – $K^* = 1$; *4* – $K^* = 2$

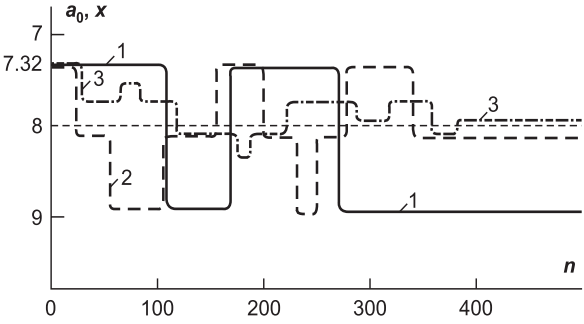


Fig. 12.5.
The investigation of the system adjustment block memory m_n influence when $\Delta a_0 = 0.25$; $K^* = 0.5$; $a_0(0) = 7.32$: *1* – $m_n = 5$; *2* – $m_n = 10$; *3* – $m_n = 20$

12.2.2
Multidimensional Case – Analytical Adjustment Algorithm

The analytical adjustment procedure in the case of multi-modal input signal distribution was investigated in the example of α_{2g} minimization in the neuron with a solution continuum (Chap. 1) and arc tangent activation function ($B = 10$).

The following problems were analyzed experimentally:

1. The influence of the initial conditions on the convergence of the iteration procedure at the search of one local extremum;

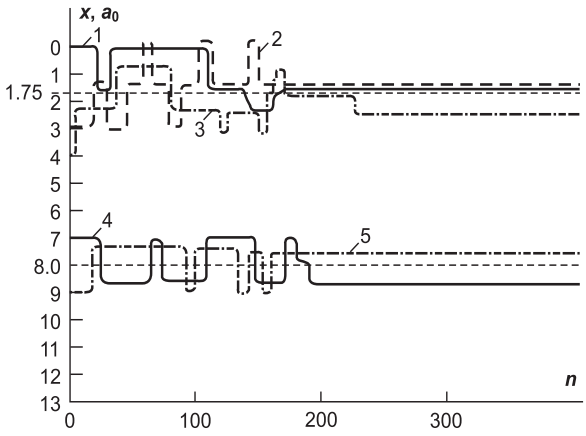


Fig. 12.6.
The investigation of the initial conditions influence upon the system adjustment dynamics when $\Delta a_0 = 0.25$; $K = 10$; $m_n = 10$: *1* – $a_0(0) = 0$; *2* – $a_0(0) = 3$; *3* – $a_0(0) = 4$; *4* – $a_0(0) = 7$; *5* – $a_0(0) = 9$

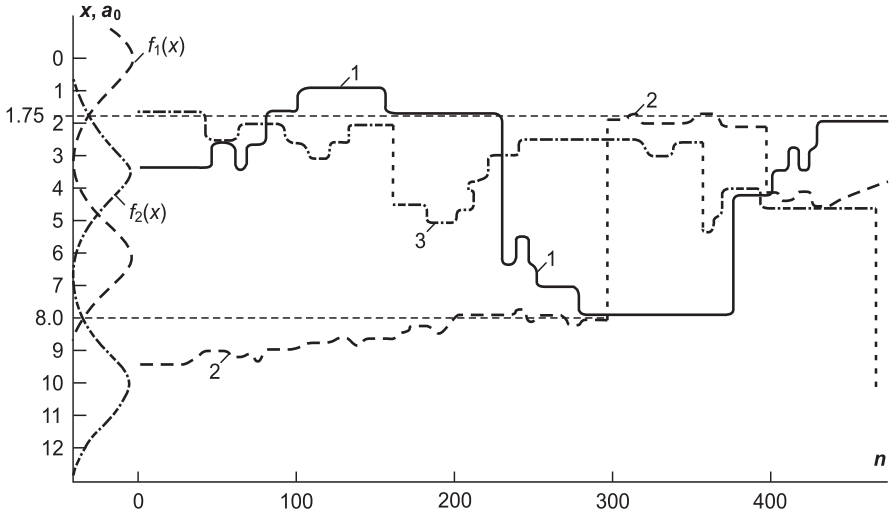


Fig. 12.7. The system adjustment dynamics under the set of random initial conditions: *1* – $\Delta a_0 = 0.25$; $K = 0.5$; $m_n = 10$; *2* – $\Delta a_0 = 0.25$; $K = 0.25$; $m_n = 20$; *3* – $\Delta a_0 = 2$; $K = 2$; $m_n = 10$

2. The influence of the step value and feature space dimensionality on the convergence rate of the iteration procedure. Stability of the gradient procedure. The influence of variance value on the quality of the iteration process convergence;
3. The influence of the gradient calculation method on the search process quality;
4. The influence of the system adjustment block memory m_n.

The investigation was carried out with the help of the random-vector generator **x** and teacher instruction ε. The multi-modal distribution of the random vectors is shown in Fig. 12.8. The circles indicate the level of equal values for probability density of each mode (solid lines – first class patterns, dashed lines – second class patterns). The whole number of mode was $Z = 10$, and root-mean-square value for one mode was $\sigma = 2$.

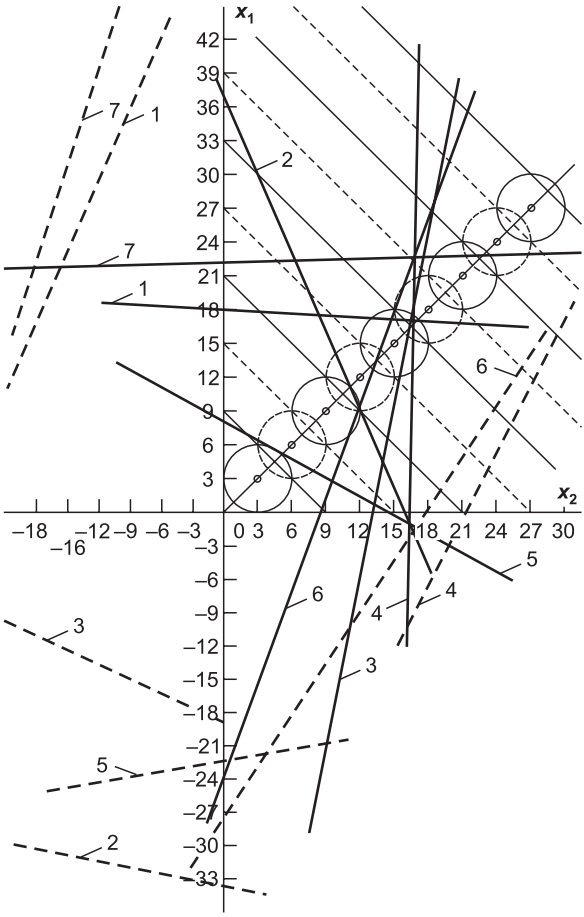


Fig. 12.8.
Illustration of the neuron operation capability under multi-modal input signal: *dashed line* indicates the initial hyperplane position; *continuous line* indicates the intermediate hyperplane position (500 iterations); *1–7* are the numbers of experiments

The first experiment was aimed at testing the stability of the neuron coefficient vector in the optimal state. For this purpose, the optimal initial conditions corresponding to one of the local extrema were given to this vector (positions *1* and *1a* in Fig. 12.9) and the learning process started. The initial (*1* and *1a*) and final (*1′* and *1′a*) hyperplane positions show the stability of the extremum corresponding to one of minimums of the average risk function. Oscillations of the relatively stable positions are the result of the stochastic properties of the minimized neural network quality functional. The deviation from the optimal position with rotational displacement *3* and without it *2* results in the hyperplane displacement into the nearest local minimum *2′* and *3′*. Figure 12.9 shows initial (*1, 2, 3, 4*) and final (*1′, 2′, 3′, 4′*) divisional hyperplane positions for different initial conditions.

Figure 2.10 shows the line (hyperplanes in the general case) coefficient adjustment dynamics under multi-modal input signal distribution. It was convenient in this case to use the intercept form of the equation of a straight line (Fig. 12.10) and trace these intercept length changes. It is seen that under the optimal initial conditions (*1, 2*), the

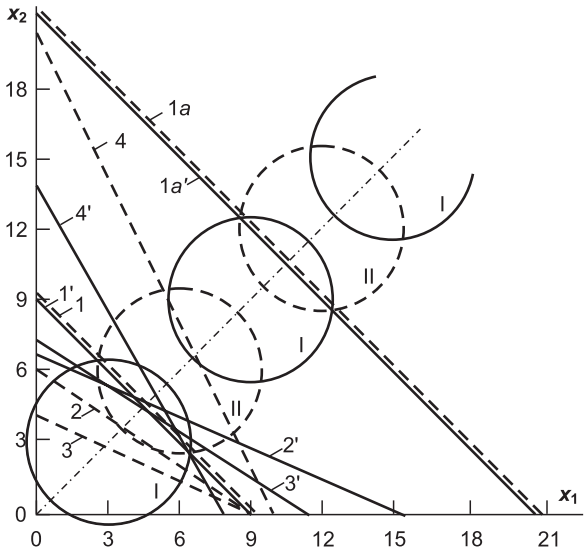


Fig 12.9.
Neuron coefficient adjustment dynamics under multi-modal input signal distribution and $m_n = 30$ (numeric characters are the numbers of experiments): *dashed lines* are the initial hyperplane positions; *continuous lines* are the final hyperplane positions

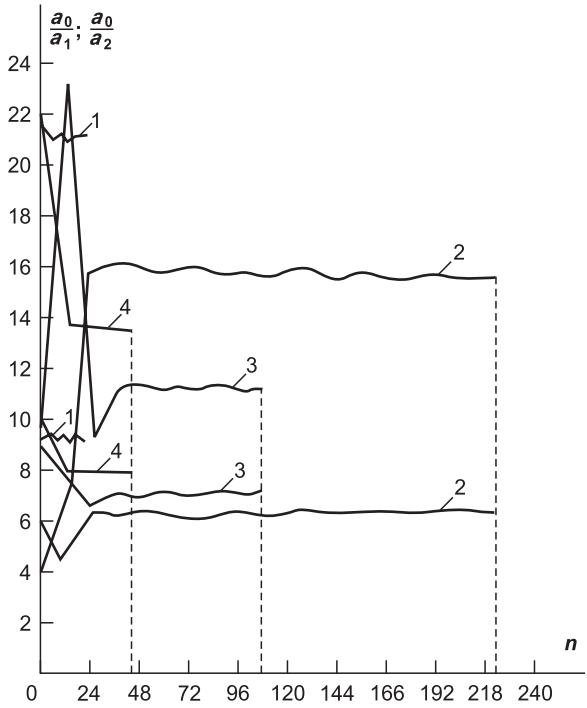


Fig. 12.10.
Neuron coefficient adjustment dynamics under multi-modal input signal distribution: *1–4* are the numbers of experiments

system oscillates slightly around the optimal position. The large oscillations of line *3* are caused by the large values of the functional gradient. This is the property of the points close to the local extremum. The points far from the local extremum are char-

acterized by the small gradient values, and therefore its movement is slow. Consequently, it is necessary to know some a priori information about the quality functional (limitation of the search space, expected characteristics of the extrema location, etc.) in order to select the proper search intervals and initial steps of the gradient procedure.

Interesting results are obtained in the investigation of the variance value (the level of class quality) on the adjustment process. If the variance is small relative to the distances between modes, then the optimal position of the divisional surface is not significant because the classes do not overlap and the local extrema are not sharp. The experiment with the first class variance being several times greater than the second class variance showed that the optimal divisional hyperplane position shifted to the mode with the smaller variance. This could be expected for the system adjusted by the average risk function.

The stability of the gradient procedure is achieved by the experimental selection of the step value and constraint upon the vector component increments. The component increments could not be more than the fourth part of the distance between local extrema, and the learning procedure was smooth.

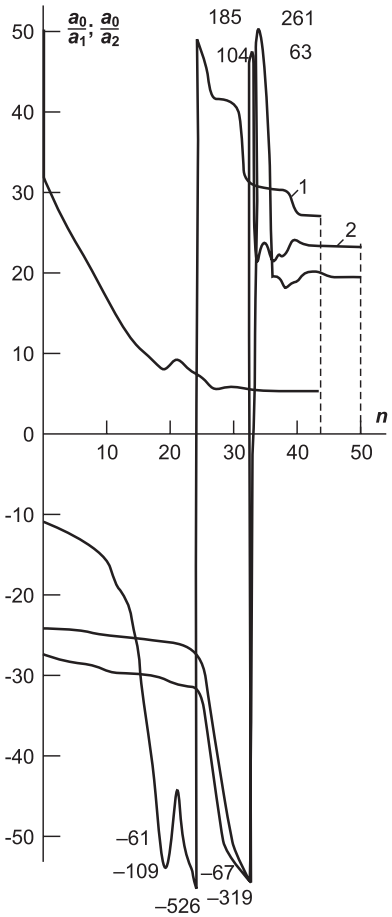


Fig. 12.11.
Neuron coefficient adjustment dynamics under four modes of the input signal distribution: *1* is the first minimum; *2* is the second minimum

Two pairs of curves for the search dynamics of two minimums under four modes of the input signal distribution are represented in Fig. 12.11. The initial optimal gradient procedure step was equal to 4, and constraint for Δa_i was equal to 0.03. Interestingly, the neuron coefficients changed the sign in the adjustment process. The image point for the first minimum search transfers from the second to the first quadrant of the adjustable coefficient space, and the image point for the second minimum search transfers from the third to the first quadrant.

12.3 Investigation of Dynamics for the Neural Networks of Particular Form for the Non-Stationary Pattern Recognition

This section deals with a one-dimensional neural network and a_{2g} minimization (Chap. 9). The aim of the investigation was the estimation of different system characteristics upon the closed-loop adjustment circuit functioning.

The expression for the analogous error of the system has the following form:

$$x_a(n\Delta T) = \varepsilon(n\Delta T) - x(n\Delta T) + a_0(n\Delta T) \tag{12.1}$$

Consequently,

$$\overline{x_a^2(n\Delta T)} = \varepsilon^2(n\Delta T) - \overline{x^2(n\Delta T)} + \overline{a_0^2(n\Delta T)} - \overline{2\varepsilon(n\Delta T)x(n\Delta T)} + \overline{2a_0(n\Delta T)\varepsilon(n\Delta T)} - \overline{2a_0(n\Delta T)x(n\Delta T)}$$

The lines over the expressions mean the averaging performed at the time instant $n\Delta T$ across the set of implementations of the nonstationary random process. Since we have only one implementation of the nonstationary random process $x_a(n\Delta T)$ then the averaging across the set must be substituted by the averaging across the time on the memory interval m_n with additional constraints according to the a priori information about the characteristics of parameter changes for nonstationary random process distribution across the memory interval. The most suitable in this case is the representation of the random process in the form of the sum of the stationary process and deterministic process with known characteristics of its changes in the functional form [12-1].

Since the derivative

$$+\overline{\frac{\mathrm{d}x_a^2(n\Delta T)}{\mathrm{d}a_0(n\Delta T)}}$$

cannot be expressed in the algebraic form then let us assume that the value $a_0(n\Delta T)$ is fixed on the averaging interval m_n. The change of $a_0(n\Delta T)$ is performed in the adaptation mode with the cycle equal to the memory m_n of the system adjustment block.

Consequently,

$$+\overline{\frac{\mathrm{d}x_a^2(n\Delta T)}{\mathrm{d}a_0(n\Delta T)}} = \overline{2x_a(n\Delta T)}$$

The closed-cycle adjustment algorithm has the following form:

$$a_0\left[\left(\frac{n}{m_n}+m_n\right)\Delta T\right]=a_0\left[\frac{n}{m_n}\Delta T\right]+K^*\overline{x_a(n\Delta T)} \tag{12.2}$$

In the case of nonstationary patterns, the process $x_a(n\Delta T)$ is a nonstationary one with characteristics determined by the input signal nonstationary characteristics (Chap. 9). The task to determine the average

$$\overline{x_a(n\Delta T)}$$

is the classical problem of filtering of nonstationary discrete random processes [12-3 to 12-7]. Methods of recurrent implementation of the optimal discrete filters [12-6] were used for the given neural network modeling.

If $m_n = m = \text{const}$, then for each n and $\Delta T = 1$ in this case

$$a_0\left(\frac{n}{m_n}+m_n\right)=a_0\left(\frac{n}{m_n}\right)+K^*\sum_{i=0}^{m}W(i,n)x_a(ni)$$

where $W(i,n)$ is the optimal impulsive admittance function of the estimation filter

$$\overline{x_a(n\Delta T)}$$

Below we use for $W(i,n)$ the expressions from [12-6, 12-7].

The dependence of the following input signal characteristics upon the dynamic of the closed-cycle adjustable system were investigated:

1. Time course of the pattern assemblage mathematical expectations (the assemblages of both classes are assumed to be equal);
2. Level of class intersection determined by the variance equal for both classes under the fixed difference between the mathematical expectations of pattern assemblages of the first and second classes);
3. Nonstationary level determined in particular by the change rate of the class centers' coordinates;
4. Memory value m_n in the block of the system closed-cycle adjustment;
5. Prediction time α in the block of the system closed-cycle adjustment at the estimation of the secondary functional gradient;
6. Amplification coefficient K^* in the block of the system closed-cycle adjustment.

Figures 12.12–12.19 show the time courses of the neuron threshold changes under the linear laws for the changes of the class centers' coordinates. Two laws with different change rates of these coordinates were used: $(2t + 3)$ and $[(1/2)t + 3]$. The distances between class centers is fixed in all the experiments.

Groups of curves I and II correspond to the two aforementioned linear laws. The data analysis results in the following conclusions:

1. The increase of the memory value m_n of the recognition system results in the decrease of the class intersection level influence upon the adjustment random error;
2. The increase of m_n results in the increase of the systematic error in the coefficient adjustment (Figs. 12.14, 12.15);

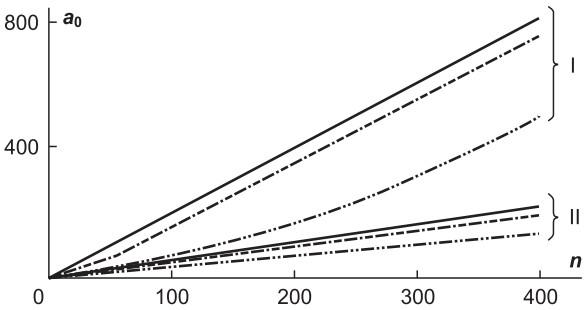


Fig. 12.12.
Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $\sigma = 3$; $\alpha = 0$; $K^* = -0.1$: —··— $m = 20$; – – · – – $m = 3$; —— ideal threshold value

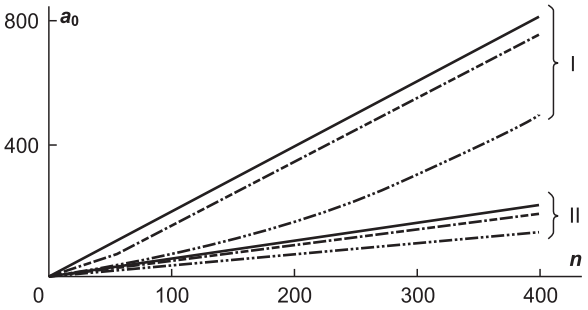


Fig. 12.13.
Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $\sigma = 10$; $\alpha = 0$; $K^* = -0.1$: —··— $m = 20$; – – · – – $m = 3$; —— ideal threshold value

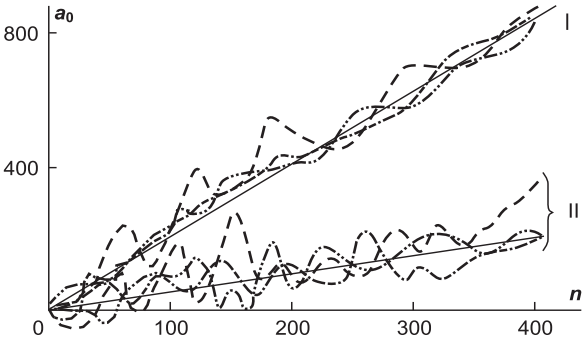


Fig. 12.14.
Dynamics of the system closed-cycle adjustment under the non-stationary pattern recognition at $K^* = -0.5$; $m = 3$: – – – – – $\alpha = 2$, $\sigma = 1$ (ideal threshold value); – – · – – $\alpha = 10$, $\sigma = 0.5$; —··— $\alpha = 20$, $\sigma = 5$; —— $\alpha = 40$, $\sigma = 5$

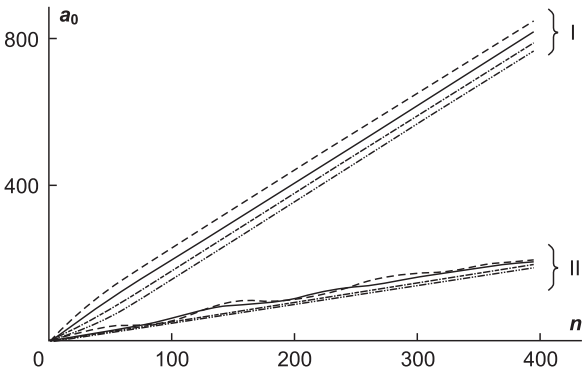


Fig. 12.15.
Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $K^* = -0.5$; $m = 20$: —··— $\sigma = 1$, $\alpha = 2$; – – · – – $\sigma = 5$, $\alpha = 10$; —— $\sigma = 5$, $\alpha = 20$; – – – – – $\sigma = 5$, $\alpha = 40$

3. The adjustment process is unstable under the small values of m_n ($m_n = 5$) and $K^* = -2$. The increase of m_n up to $m_n = 20$ results in the stable adjustment process. Consequently, the nonstationary pattern recognition systems require the memory in the adjustment block ($m_n > 1$). The increase of m_n compensates to some extent for the lack of the a priori information about K^*:
4. The rate of the classes' center coordinate changes in time does not practically influence the errors of the adjustment circuit performance;
5. The requirement $K^* < -1$ is necessary for the stability of the adjustment circuit;
6. The characteristic modulation of the enveloping curve for the system threshold changes under the unstable conditions is observed;
7. The results of the use of the quadratic law instead of the linear law for the classes' centers' coordinate changes in time confirm the previous conclusions (p. 1–6). Under the sufficiently large values of m_n, the systematic error changes of the adjustment circuit show some regular relationship (negative for $K^* > -1$ and positive for $K^* < -1$).

Figures 12.17–12.19 show that the level of class intersection significantly influences the adjustment circuit in the self-oscillatory adjustment process when $K^* = -2$. The adjustment process diverges under the large values of σ. But under the small σ, the oscillating adjustment process periodically changes its amplitude relatively ideal threshold value and at some time moments become rather precise.

The experiments with solution prediction on the time interval α (Figs. 12.14, 12.15) showed the following:

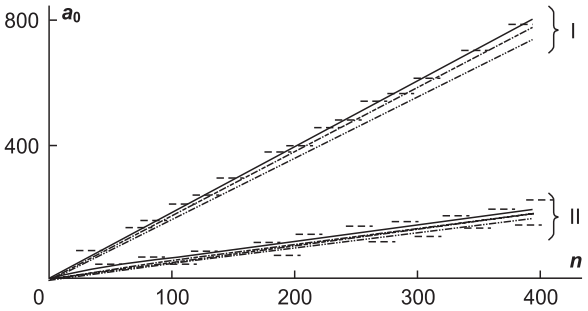


Fig. 12.16.
Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $\sigma = 5$; $\alpha = 0$; $m = 20$: —··— $K^* = -0.5$; – – · – – $K^* = -0.75$; —— $K^* = -2$ (ideal threshold value)

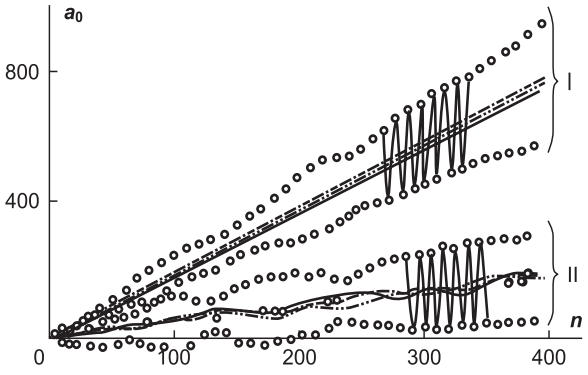


Fig. 12.17.
Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $\sigma = 5$; $\alpha = 0$; $m = 5$: —··— $K^* = -0.5$; – – · – – $K^* = -0.75$; —— $K^* = -1$ (ideal threshold value); ◦◦◦◦◦ $K^* = -2$

1. The increase of α results in the increase of the random error in the closed-cycle adjustment circuit;
2. The increase of σ, decrease of m, and fixed α result in the increase of the random error in the closed-cycle adjustment circuit;
3. The results of comparison between the linear and quadratic laws for the classes' centers' coordinate changes in time showed that the random error in the closed-cycle adjustment circuit increased in the former case.

12.4 Dynamics of the Three-Layer Neural Network in the Learning Mode

The considered neural network is supposed to have a continuum of solutions. The first, second, and third neural network layers consisted of 3, 2, and 1 neurons, respectively. The feature space was multidimensional in the general case and was two-dimensional in the particular case. The open-loop neural network was described by the following expression:

$$x^3_{h_3} = \frac{2}{\pi}\operatorname{arctg} B \sum_{h_2=1}^{H_2} a_{h_3 h_2} \frac{2}{\pi}\operatorname{arctg} B \sum_{h_1=1}^{H_1} a_{h_2 h_1} \frac{2}{\pi}\operatorname{arctg} B \sum_{h_0=0}^{H_0} a_{h_1 h_0} x^0_{h_0} \tag{12.3}$$

$\overrightarrow{g^3_{h_3}}$ $\overrightarrow{x^2_{h_2}}$ $\overrightarrow{g^2_{h_2}}$ $\overrightarrow{x^1_{h_1}}$ $\overrightarrow{g^1_{h_1}}$

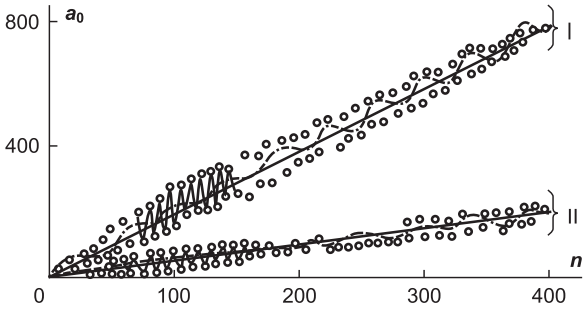


Fig. 12.18. Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $\sigma = 3$; $\alpha = 0$; $K^* = -2$: – – · – – $m = 20$; —— ideal threshold value; ◦◦◦◦◦ $m = 5$

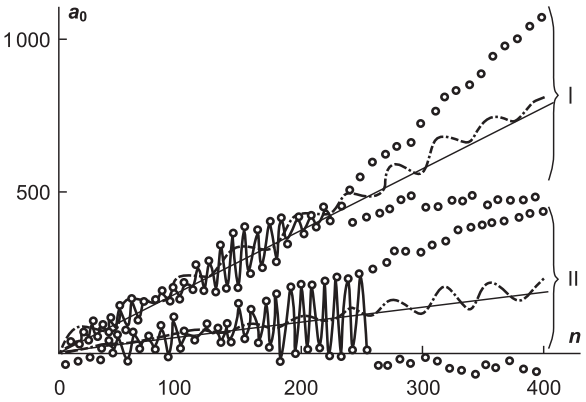


Fig. 12.19. Dynamics of the system closed-cycle adjustment under the nonstationary pattern recognition at $\sigma = 10$; $\alpha = 0$; $K^* = -2$: – – · – – $m = 20$; —— ideal threshold value; ◦◦◦◦◦ $m = 5$

The expressions for the α_{2g} gradients estimations have the following form:

$$\overline{\frac{\partial x_g^2}{\partial a_{h_1 h_0}}}^{m_n} = \frac{16}{\pi^3}\frac{B^3}{m_n}\sum_{i=1}^{m_n} x_g(i)\sum_{h_2}^{H_2}\frac{a_{h_3 h_2}a_{h_2 h_1}x_{h_0}(i)}{\left(1+\left[g_{h_3}^3(i)\right]^2\right)\left(1+\left[g_{h_2}^2(i)\right]^2\right)\left(1+\left[g_{h_1}^1(i)\right]^2\right)}$$

$$\overline{\frac{\partial x_g^2}{\partial a_{h_2 h_1}}}^{m_n} = \frac{8}{\pi^2}\frac{B^2}{m_n}\sum_{i=1}^{m_n} x_g(i)\frac{a_{h_3 h_2}x_{h_1}^1(i)}{\left(1+\left[g_{h_3}^3(i)\right]^2\right)\left(1+\left[g_{h_2}^2(i)\right]^2\right)}$$

$$\overline{\frac{\partial x_g^2}{\partial a_{h_3 h_2}}}^{m_n} = \frac{4}{\pi}\frac{B}{m_n}\sum_{i=1}^{m_n} x_g(i)\frac{x_{h_2}^2(i)}{1+\left[g_{h_3}^3(i)\right]^2}$$

The equal value lines for the first and second pattern class distributions are represented in Fig. 12.20. The state of the neural network is determined in the following way. Three neurons of the first layer have respectively the coefficients $a_{10} = -12, a_{11} = 1, a_{12} = 1$, $a_{20} = 24$, $a_{21} = -1$, $a_{22} = -1$, $a_{30} = -36$, $a_{31} = 1$, and $a_{32} = 1$. Neurons of the second layer have the coefficients $a'_{10} = 0, a'_{11} = 1, a'_{12} = 1, a'_{13} = 1, a'_{20} = 0, a'_{21} = 1, a'_{22} = 1$, and $a'_{23} = 1$. Neurons of the third layer have the coefficients $a''_{10} = 0$, $a''_{10} = 1$, and $a''_{12} = 1$.

In the experiments, the input feature space dimensionality was $N = 2$, and the number of modes $f(x)$ was 4.

Experiments with the first neuron layer (the second and the third layers are optimal). The conditions in each experiment were the following:

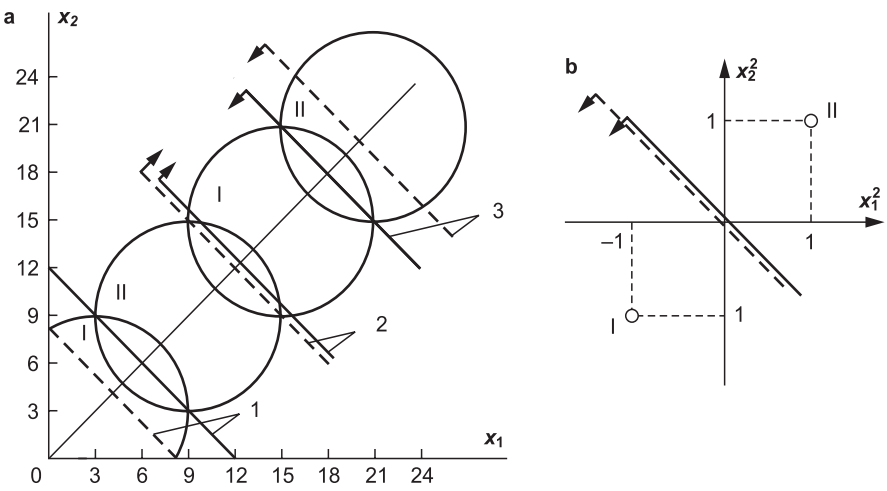


Fig. 12.20. Initial and final positions of divisional surfaces realized by the neurons in the experiment 1–3: **a** the first layer; **b** the second layer

1-1. The aforementioned hyperplane coefficients are optimal;
1-2. Parallel shift of hyperplanes realized by the neurons of the first neural network layer (the initial coefficient: 1, 1, –8, –1, –1.2; 1.1, –32);
1-3. Parallel shift of two hyperplanes realized by the neurons of the first neural network layer in different directions (the initial coefficient: 1.1, –8, –1, –1.24; 1.1, –40).

Experiments with the second neuron layer (the first and the third layers are optimal).

2-1. Rotation of one hyperplane realized by the neurons of the second neural network layer by the angle $\alpha = \pi$ (the initial coefficient: –1, –1, –1, 1.1, 1);
2-2. Rotation of two hyperplanes realized by the neurons of the second neural network layer by the angle $\alpha = \pi$.

Experiments with the third neuron layer (the first and the second layers are optimal).

3-1. Rotation of the hyperplane realized by the neuron of the third neural network layer by the angle $\alpha = \pi$.

The following results were obtained.

Figures 12.21–12.23 illustrate the coefficient adjustment procedure. The vertical axis represents the coefficient values, and the horizontal one represents the number of iterations. The coordinate axis level corresponds to the optimal coefficient ratios. Experiment 1-1 confirms the stability of coefficient values in the optimal state (small deviations of their values from the optimal one at the sufficiently large number of iterations). In experiments 1-2 and 1-3, the gradient procedure provides such an adjustment that the divisional surfaces reach the optimal position after 25–30 iterations.

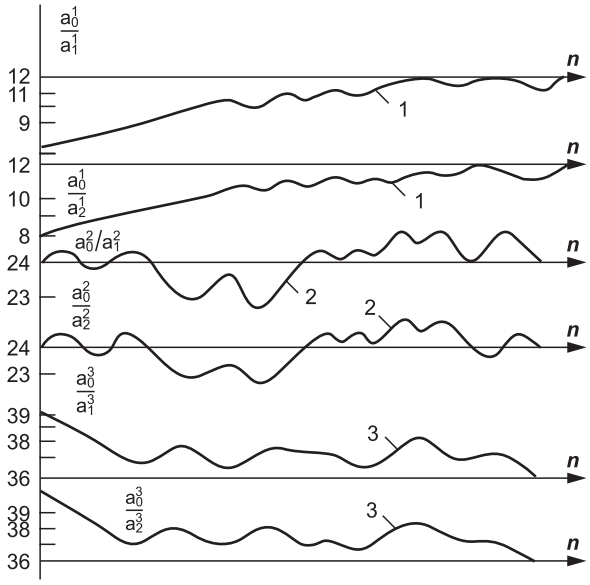


Fig. 12.21.
Coefficient adjustment dynamics in experiment 1–2 (the number of iterations is 50) at $m_n = 50$; $K_1^* = 0.1$; $K_1 = 0.01$; $K_2 = 0.1$;and $K_3 = 0.1$; K_1, K_2, K_3 are the weighting coefficients to K^* for the neurons of the first, second, and third layers: *1* – the first neuron; *2* – the second neuron; *3* – the third neuron

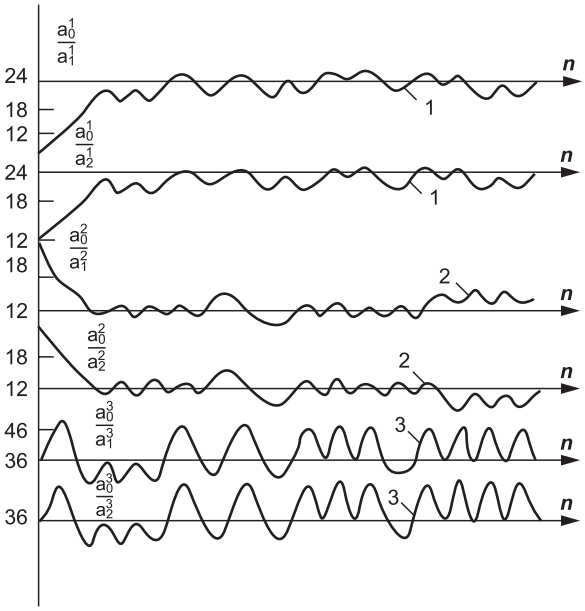


Fig. 12.22. Coefficient adjustment dynamics in experiment 1–4 (the number of iterations is 32) at $m_n = 50$; $K_1^* = 1$; $K_1 = 0.05$; $K_2 = 0.05$; and $K_3 = 0.05$: *1* – the first neuron; *2* – the second neuron; *3* – the third neuron

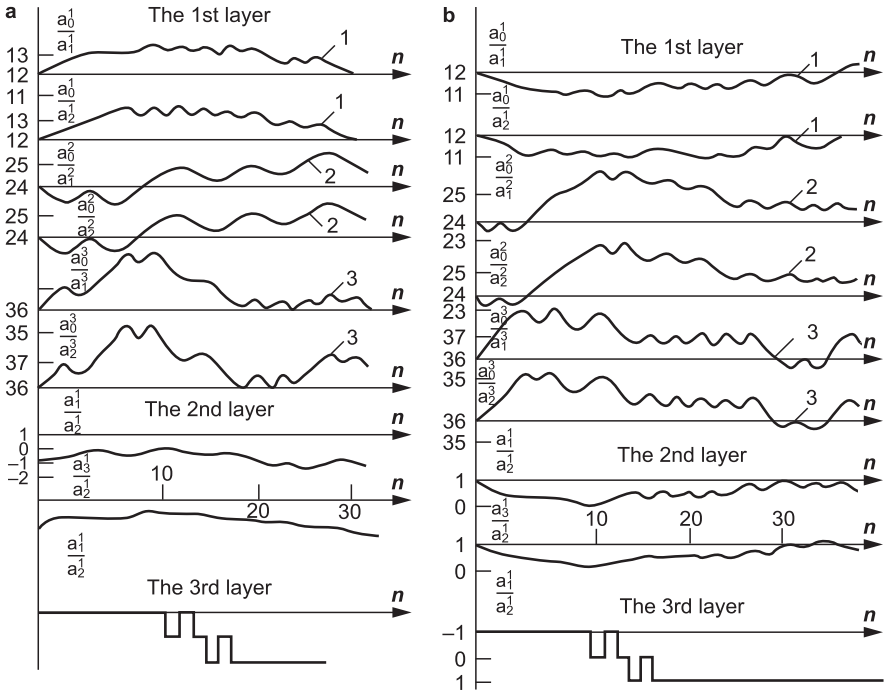


Fig. 12.23. Coefficient adjustment dynamics in the experiment: **a** experiment 2-2; **b** experiment 3.1; *1* – the first neuron; *2* – the second neuron; *3* – the third neuron

The results of experiment 1-4 are rather interesting. The initial conditions provided the optimal positions of the surfaces realized by the neurons of the first, second, and third layers in such a way that the "inverse" classification took place. The adjustment resulted in the parallel shifts of the planes, though the rotation by 180° is also possible. The divisional planes realized by the neurons of the second and third layers are drawn through the origin of coordinates, i.e., only the rotation of these planes around the origin of coordinates is possible in the adjustment procedure. Therefore, the experiments with the neurons of these layers included the adjustment of coefficients for the planes turned through 180°.

Figure 12.23 shows coefficient adjustment dynamics in experiments 2-2 and 3-1. The adjustment procedure resulted in the turn of the hyperplane realized by the neurons of the third layer through 180°, thus taking an optimal position. After the adjustment of the third layer neurons, the coefficients of the first layer neurons also take the optimal values.

The results of the performed experiments confirmed the theoretical analysis of the investigated adjustment algorithm and demonstrated its high efficiency. The following wide range of problems remains to be analyzed:

1. Selection of the optimal coefficients K in the gradient procedure and their relationship between multilayer neural network layers;
2. Analysis of the influence of the multilayer neural network structure redundancy upon the adjustment quality.

12.5 Investigation of the Particular Neural Network with Backward Connections

We consider here a one-dimensional neuron with the feedback described by the following relationships:

$$y(n\Delta T)=\operatorname{sign}\left[g(n\Delta T)\right];\quad g(n\Delta T)=x(n\Delta T)-a_0(n\Delta T)+a_k(n\Delta T)x_k\left[(n-1)\Delta T\right] \tag{12.4}$$

where ΔT is the time interval between the presentation of the input patterns. The minimum α_{2a} criterion is taken as the criterion of the secondary optimization. It is assumed that the coefficients α_0 and α_k do not change in the interval of averaging m_n during the closed-cycle adjustment. The expressions for the α_{2a} gradient estimation have the following form:

$$\frac{\overline{\partial x_a^2(n\Delta T)}^{m_n}}{\partial a_0}=2\overline{x_a(n\Delta T)}^{m_n}$$

$$\frac{\partial x_a^2(n\Delta T)}{\partial a_k}=-2\overline{x_a y\left[(n-1)\Delta T\right]}^{m_n} \tag{12.5}$$

The expressions (12.4) and (12.5) form the base for the design of the corresponding closed system. The averaging of the gradient measurements was performed with the help of the optimal discrete filter. The a priori hypothesis about the change of the input signal mathematical expectation (stationary, linear, quadratic, etc.) was used for its synthesis.

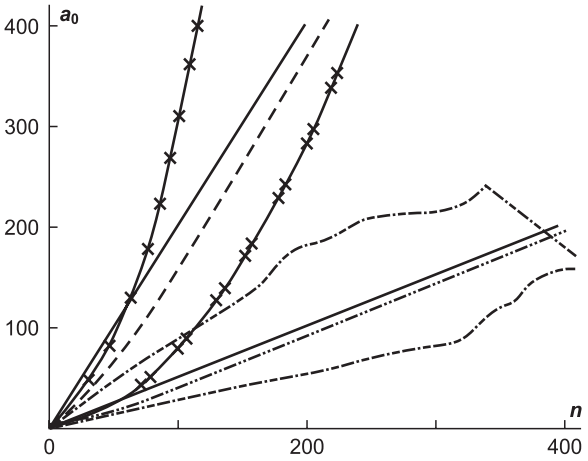


Fig. 12.24.
Dynamics of adjustment of the neuron with feedback ($m = 20$; $K^* = -0.5$): —— ideal case ($a_0 = 2t + 3$); – – – – – without feedback; – – · – – positive feedback at $K_1^* = 0.5$ and negative feedback at $K_1^* = 0.5$; —×— positive feedback at $K_1^* = -0.5$; – – – – – ideal case ($a_0 = 0.5t + 3$); —··— without feedback and positive feedback at $K_1^* = 0.5$; —·— negative feedback at $K_1^* = -0.5$; —××— positive feedback at $K_1^* = -0.5$

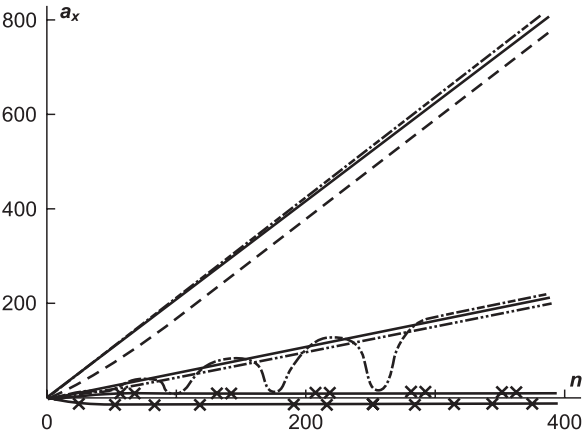


Fig. 12.25.
Dynamics of adjustment of the neuron with feedback: —— ideal case ($a_0 = 2t + 3$); – – – – – without feedback; – – · – – –positive feedback at $K_1^* = 0.5$ and negative feedback at $K_1^* = 0.5$; —×— positive feedback at $K_1^* = -0.5$; —— ideal case ($a_0 = 0.5t + 3$); —··— without feedback; – – · – – positive feedback at $K_1^* = 0.5$ and negative feedback at $K_1^* = -0.5$; —××— positive feedback at $K_1^* = -0.5$

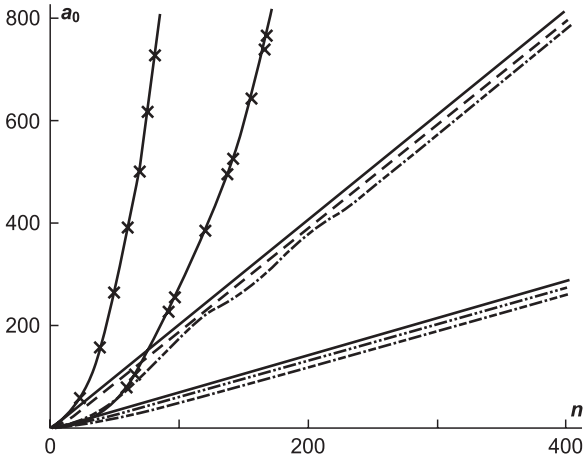


Fig. 12.26.
Dynamics of adjustment of the neuron with feedback ($m = 5$; $K^* = -0.5$): —— ideal case ($a_0 = 2t + 3$); – – – – – without feedback; – – · – – positive feedback at $K_1^* = 0.5$ and negative feedback at $K_1^* = -0.5$; —×— positive feedback at $K_1^* = -0.5$; —··— without feedback; – – · – – negative feedback at $K_1^* = -0.5$ and positive feedback at $K_1^* = -0.5$; —××— positive feedback at $K_1^* = -0.5$; —— ideal case ($a_0 = 2t + 3$)

The analysis of the experimental study for the neuron with feedback partially represented in Figs. 12.24–12.29 shows the following:

1. The introduction of the positive or negative feedback into the open-loop system ($K^* > 0$ or $K^* < 0$ in the coefficient a_k adjustment circuit) gives the same results of the system adjustment for the value a_0 and total threshold $a_\Sigma = a_0(n) + a_k(n)x_k(n-1)$, but oppositely signed and equal by the absolute value coefficients a_k:
2. At the sufficiently large value of the system memory in the adjustment block (about $m_n = 20$), the change of a_0 and a_k has the form of oscillations. At the decrease of the memory value (to about $m_n = 5$), the oscillations of the adjustment process for the coefficient a_k sharply increase, and for the coefficient a_0 – decrease;

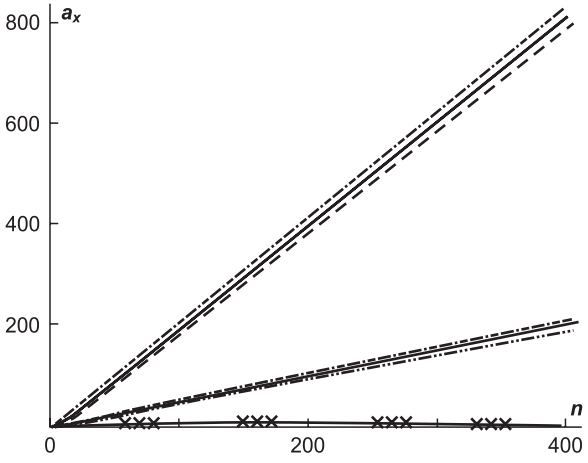


Fig. 12.27.
Dynamics of adjustment of the neuron with feedback: —— ideal case ($a_0 = 2t + 3$); – – – – – without feedback; – – · – – positive feedback at $K_1^* = 0.5$ and negative feedback at $K_1^* = -0.5$; —— ideal case ($a_0 = 0.5t + 3$); — ·· — without feedback; – – · – – positive feedback at $K_1^* = -0.5$ and negative feedback at $K_1^* = -0.5$; — ××× — positive feedback at $K_1^* = -0.5$

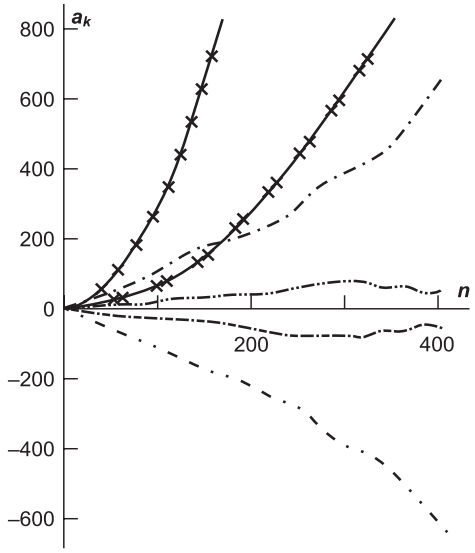


Fig. 12.28.
Dynamics of adjustment of the neuron with feedback ($m = 20$; $K^* = -0.5$). *For the ideal case* ($a_0 = 2t + 3$): — × — positive feedback at $K_1^* = -0.5$; — · — negative feedback at $K_1^* = 0.5$; – – ·· – – positive feedback at $K_1^* = -0.5$; *For the ideal case* ($a_0 = 0.5t + 3$): — ×× — positive feedback at $K_1^* = -0.5$; – – · – – positive feedback at $K_1^* = 0.5$; — ·· — negative feedback at $K_1^* = -0.5$

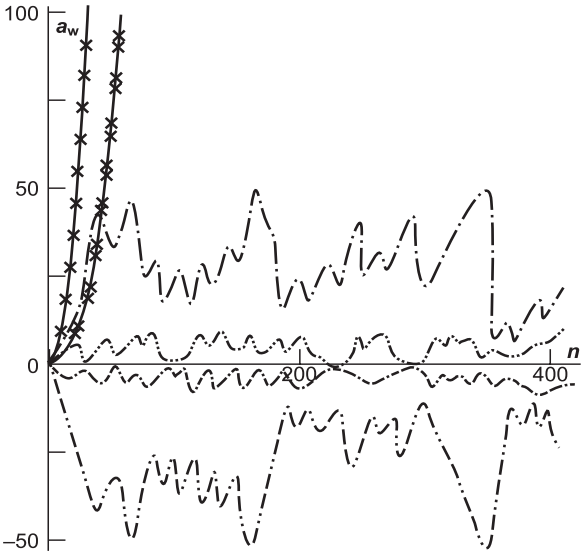


Fig. 12.29. Dynamics of adjustment of the neuron with feedback ($m = 5$; $K^* = -0.5$). *For the ideal case* ($a_0 = 2t + 3$): —×— positive feedback at $K_1^* = -0.5$; —·— negative feedback at $K_1^* = -0.5$; – – ·· – – positive feedback at $K_1^* = 0.5$; *For the ideal case* ($a_0 = 0.5t + 3$): —××— positive feedback at $K_1^* = -0.5$; —··— negative feedback at $K_1^* = -0.5$; – – ·· – – positive feedback at $K_1^* = 0.5$

3. The systematic error of the coefficient a_0 adjustment increases as the system memory increases or at the introduction of the feedback into the open-loop structure of the system. The adjustment systematic error for the total neuron threshold is practically zero. This is an advantage of the neural network with feedback over the neural network without feedback;
4. The decrease of the coefficient K_1^* in the iteration procedure for the adjustment of the feedback gain results in the approximation between characteristics of the neuron with the feedback to the neuron without feedback.

12.6 Dynamics of One-Layer Neural Networks in the Learning Mode

Three types of the neural networks in the self-learning mode are considered below: the neural network with the search of the distribution mode centers $f(\mathbf{x})$; the neural network in the form of the neuron layer with two solutions; the neural network in the form of the neuron with K_p solutions.

The main goal of the study is the quality estimation of the developed algorithms when the input signal $\mathbf{x}(n)$ has the arbitrary modality distribution.

12.6.1 Neural Network with the Search of the Distribution Mode Centers *f*(*x*)

The self-learning algorithm realizing the following recurrent relationship (Chap. 9)

$$\mathbf{b}(x_k, n+1) = \mathbf{b}(x_k, n) + K^* \left[x(n) - \mathbf{b}(x_k, n) \right] \tag{12.6}$$

is considered. The algorithm consists of the following stages:

1. Coordinates of the K_p-dimensional vector $\mathbf{b}(x_k,0)$ are randomly selected in the given interval of variation of x;
2. The successive pattern x is presented to the input. The center $\mathbf{b}$ closest to $\mathbf{x}$ is calculated;
3. This coordinate of vector $\mathbf{b}(x_k)$ is changed according to (12.6);
4. The internal cycle is locked on p. 2. Then the external cycle is locked on p. 1. The random input signal $x(n)$ distribution is the sum of the normal distributions with the given variances and mathematical expectations equal to $2, 4, \ldots, 16$. The number of distribution modes $f(x)$ was fixed from 2 to 8.

Figure 12.30 shows the typical change of class center coordinates $\mathbf{b}(x_k,n)$ in the adjustment process for some variant of the random initial conditions. The algorithm functioning results are represented in Tables 12.1 and 12.2. In the tables, i is the number of mode in the distribution $f(x)$, Z is the modality of the distribution $f(x)$, and j is the number of cycle for the injection of the random initial conditions $\mathbf{b}(x_k,0)$ in the extremum R search. Table 12.1 was calculated at $K_p = 5$, $M = 300$ (the number of iterations by n), and $K^* = 0.02$. For each σ here, the number of distribution $f(x)$ modes found for each j is given in the right column. The number of modes found during all the previous cycles is given in the left column. The similar data are represented in Table 12.2 for $Z = K_p$, $K^* = 0.01$, $M = 100$, and $\sigma = 0.5$. The analysis of the obtained results shows the following:

1. The algorithm is efficient at the solution of a rather complex self-learning task;
2. The experimental results confirm theoretical conclusions (Chap. 9) concerning the expected local and global function extremum search;
3. The increase of σ results in the decrease of the algorithm functioning quality under the fixed Z, K_p, K^*, i, j.

The considered algorithm was slightly modified (the set of initial conditions in terms of class center coordinates was substituted by the set in terms of coordinates of the initially presented Z patterns). The respective quality increase is illustrated by the data shown in Table 12.3. Here the number of distribution modes obtained at each A-th step of the initial condition injections and the total number of distribution modes obtained during the total number of A steps are represented. The search was performed using both modifications of the described algorithm. The experiment parameters were the

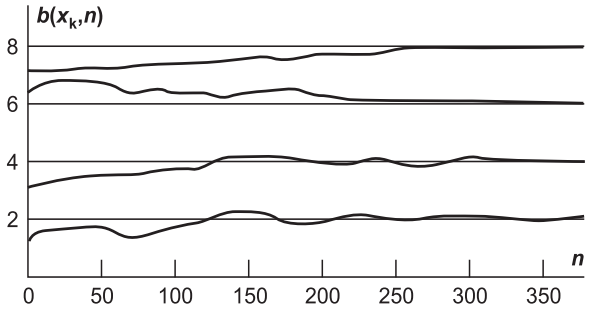


Fig. 12.30. Typical change of class center coordinates in the adjustment process at $K^* = 0.02$; $\sigma = 0.5$

Table 12.1.
Algorithm functioning results; $K_p = 5$, $M = 300$, and $K^* = 0.02$

i	Z	j	$\sigma = 0.1$		$\sigma = 0.3$		$\sigma = 0.5$		$\sigma = 0.7$		$\sigma = 0.9$	
1		1	2	2	2	2	2	2	2	2	0	0
2		2	2	2	2	2	2	1	2	1	0	0
	2	3	2	2	2	0	2	1	2	0	0	0
		4	2	1	2	2	2	2	2	0	0	0
		5	2	0	2	0	2	0	2	0	1	1
1		1	3	3	1	1	1	1	1	1	1	1
2		2	3	2	3	3	1	0	1	0	1	0
3	3	3	3	3	3	2	2	2	2	2	2	1
		4	3	3	3	3	3	3	2	1	2	2
		5	3	2	3	2	3	1	2	2	2	1
1		1	2	2	2	2	1	1	1	1	2	2
2		2	4	4	3	3	2	1	2	1	2	1
3	4	3	4	2	3	1	2	1	2	1	2	2
4		4	4	2	4	3	3	1	3	1	2	0
		5	4	4	4	4	4	3	3	2	2	1
1		1	3	3	2	2	0	0	0	0	0	0
2		2	4	2	3	1	1	1	1	1	2	2
3	5	3	5	3	5	3	2	2	3	2	4	3
4		4	5	2	5	1	4	1	4	2	5	2
		5	5	1	5	1	5	1	4	1	5	1

Table 12.2.
Algorithm functioning results; $Z = K_p$, $K^* = 0.01$, $M = 100$, and $\sigma = 0.5$

j	$i = 1,2,3,4,5,6$		$i = 1,2,34,5,6,7$		$i = 1,2,3,4,5,6,7,8$	
	$Z = K_p = 6$		$Z = K_p = 7$		$Z = K_p = 8$	
1	0	0	3	3	2	2
2	2	2	4	1	4	3
3	2	1	6	2	4	1
4	4	2	6	2	5	1
5	4	0	7	3	6	2
6	4	1	7	0	6	1
7	4	0	7	0	6	2
8	4	2	7	2	7	4
9	5	1	7	1	8	3
10	5	3	7	3	8	2
11	5	1	7	0	8	2
12	5	1	7	0	8	2
13	5	2	7	0	8	0
14	5	0	7	1	8	3
15	6	2	7	2	8	2
16	6	2	7	5	8	1
17	6	0	7	1	8	1
18	6	1	7	2	8	3
19	6	2	7	0	8	0
20	6	1	7	2	8	2

following: $K^* = 0.02$, $\sigma = 0.5$, $f(x)$ coefficients b_1–b_8 amounted respectively to –9.1, –7, –3 ,–5, –1, 3.13. The X space was the interval [–11, 5].

Table 12.3. Calculation with modified algorithm

Modes *i*	Number of modes *Z*	Number of experiment *j*	Selection of initial conditions			
			Equally probable		*X*(1) … *X*(*Z*)	
1-4	4	1 2 3 4 5	0 1 2 0 1	0 1 2 2 2	3 3 3 3 2	3 3 4 4 4
1-5	5	1 2 3 4 5	2 1 1 1 1	2 2 2 3 3	1 5 1 1 3	1 5 5 5 5
1-6	5	1 2 3 4 5	1 3 0 2 0	1 4 4 5 5	4 0 2 3 1	4 4 5 6 6
1-7	7	1 2 3 4 5	1 3 2 1 4	1 3 4 4 5	4 1 1 1 3	4 4 5 6 6
1-8	8	1 2 3 4 5	4 1 3 3 3	4 4 4 4 5	2 4 2 2 2	2 6 7 8 8

12.6.2 Neural Network with N^* Output Channels

We consider here the neural network in the form of the neuron layer with characteristics $N = 1$, $N^* = 3$.

Figure 12.31 represents the structure of this system. In this case,

$$y' = \left\{x - b\left[y_1(a_{01}), y_2(a_{02}), y_3(a_{03})\right]\right\}^2 \tag{12.7}$$

where the value $b(y)$ is unambiguously determined by the current values of a_{01}, a_{02}, a_{03} according to Fig. 12.32, Table 12.4, and the following expressions:

$$b_1 = a_{01} - \frac{a_{02} - a_{01}}{2}$$

$$b_i = a_{0,i-1} + \frac{a_{0i} - a_{0,i-1}}{2} \quad ; \quad i = 2, 3$$

$$b_4 = a_{03} - \frac{a_{03} - a_{02}}{2}$$

It follows from (12.7) that

$$\frac{\partial y'}{\partial a_{oi}} = -2\left[x(n) - b(\mathbf{x}, n)\right]\frac{\partial b}{\partial y_i}\frac{\partial y_i}{\partial a_{oi}} \tag{12.8}$$

where $\partial y_i/\partial a_{0i} = -1$ according to Fig. 12.31. Vector $\partial b/\partial y_i$ is calculated in the following way. Table 12.4 can be represented in the form of Table 12.5.

Consequently,

$$\left[\frac{\partial b(y)}{\partial y_1}, \frac{\partial b(y)}{\partial y_2}, \frac{\partial b(y)}{\partial y_3}\right] = \left[b_2 - b_1,\ b_3 - b_2,\ b_4 - b_3\right]$$

The final algorithm for the coefficient adjustment has the following form:

$$a_{0i}(n+1) = a_{0i}(n) + K_{0i}^{*}\left[x(n) - b(y,n)\right]\left.\frac{\partial b(y)}{\partial y_i}\right|_{y=y(n)} \tag{12.9}$$

Vector $\mathbf{b}(y,n)$ is calculated either according to the expressions described above or (in the case of the more complex structure) according to the recurrent expression

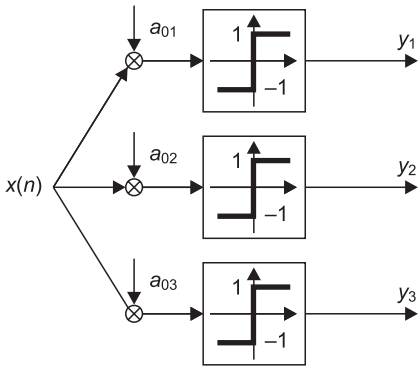


Fig. 12.31.
Block diagram of the neuron layer ($N = 1$)

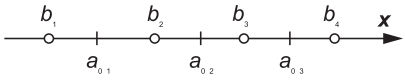


Fig. 12.32.
To the class center coordinate calculation

Table 12.4.
Values of $b(y)$ of a neural network

y	$b(y)$			
	b_1	b_2	b_3	b_4
y_1	−1	+1	+1	+1
y_2	−1	−1	+1	+1
y_3	−1	−1	−1	+1

described in the previous section. The experimentally investigated algorithm consists of the following stages:

1. Pattern $x(n)$ enters the system input;
2. The initial values of the adjustable parameters a_{0i} ($i = 1,2,3$) are selected in a random way in the interval of x variation;
3. The values $b_1, \ldots, b_4$ are calculated by the values a_{0i};
4. The component of vector $b(y)$ that is closest to $x(n)$ is selected;
5. The corresponding value is selected from vector $\partial b/\partial y_i$;
6. The system coefficients are adjusted according to p. 1,2,4,5 and expression (12.9);
7. Pattern $x(n + 1)$ enters the system input, and the adjustment process continues starting from p. 3.

Figure 12.33 shows the illustration of the system coefficient adjustment dynamics under some initial conditions (modes coordinates are 3, 5, 7; solid line – one variant, dashed line – another variant).

Table 12.5.
Data from Table 12.4 in a modified representation

y_1	y_2, y_3			
	−1, 1	**−1, 1**	**1, −1**	**1, 1**
−1	b_1	–	–	–
1	b_2	–	b_3	b_4
y_2	y_1, y_3			
	−1, 1	**−1, 1**	**1, −1**	**1, 1**
−1	b_1	–	b_2	–
1	–	–	b_3	b_4
y_3	y_1, y_2			
	−1, 1	**−1, 1**	**1, −1**	**1, 1**
−1	b_1	–	b_2	b_3
1	–	–	–	b_4

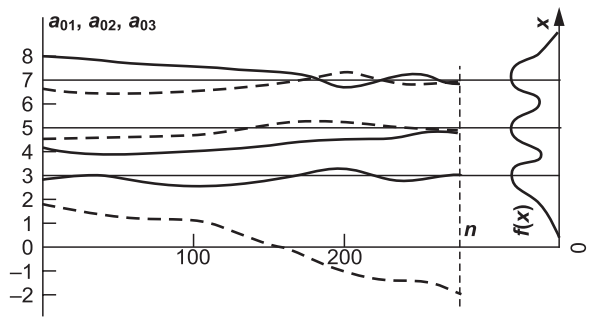


Fig. 12.33.
Dynamics of coefficient adjustment for the system represented in Fig. 12.31

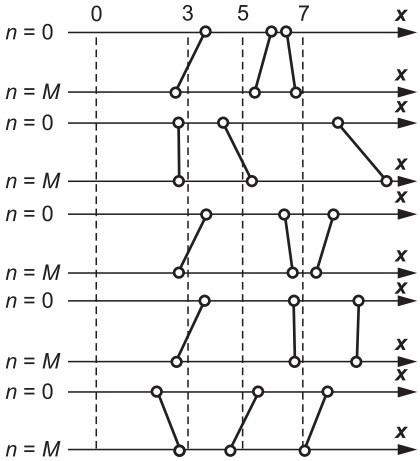


Fig. 12.34.
Results obtained under the different initial conditions and the same input sample length M

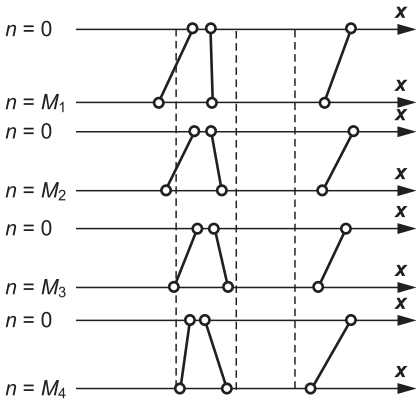


Fig. 12.35.
Results obtained under the same initial conditions and different input sample length M: $M_1 = 150$; $M_2 = 300$; $M_3 = 450$; $M_4 = 600$

Figures 12.34 and 12.35 show some results of the algorithm performance obtained under the different initial conditions and the same input sample and some results obtained under the same initial conditions and input samples of different lengths. Thick lines link the classes' center coefficients at the adjustment process start ($n = 0$) and finish ($n = M$).

12.6.3 Neuron with K_p Solutions

In this case,

$$y = 1 + \frac{1}{2}\sum_{j=1}^{K_p - 1}\left[\text{sign}\left(g - a_{j,j+1}\right) + 1\right]$$

$$g(n) = a_1(n)x(n) - a_0(n)$$

Similar to the previous case,

$$y' = [x - b(y)]^2 \quad , \quad \frac{\partial y'}{\partial a} = -2[x - b(y)]\frac{\partial b(y)}{\partial y}\frac{\partial y}{\partial a}$$

Using the expressions for $\partial y/\partial a$ (Chap. 9), one can obtain the recurrent relationships for the design of the corresponding system adjustment algorithm in the learning mode:

$$a_0(n+1) = a_0(n) - K_0^*[x(n) - b(y)]\frac{\partial b(y)}{\partial y} \quad , \quad K_0^* > 0$$

$$a_1(n+1) = a_1(n) - K_1^*[x(n) - b(y)]\frac{\partial b(y)}{\partial y}\operatorname{sign} x$$

The adjustment algorithm consists of the following stages:

1. The initial values of the adjustable parameters a_0 and a_1 are selected in a random way in some given interval;
2. The current threshold values are calculated according to the open-loop system structure and coefficients a_0 and a_1:

$$x_{j,j+1}(n) = \frac{a_{j,j+1} + a_0(n)}{a_1(n)}$$

3. The values $b(y)$ are calculated according to the expressions

$$b_1 = x_{1,2} - \frac{x_{2,3} - x_{1,2}}{2}$$

$$b_j = \left(x_{j-1,j} + x_{j,j+1}\right)\frac{1}{2} \quad , \quad j = 2,3,\ldots,K_\text{p} - 1$$

$$b_{K_\text{p}} = x_{K_\text{p}-1,K_\text{p}} + \frac{x_{K_\text{p}-1,K_\text{p}} - x_{K_\text{p}-2,K_\text{p}-1}}{2}$$

4. Pattern $x(n+1)$ enters the system input at the time instant n, and the y value is calculated;
5. The corresponding values $b(y)$ and $\partial b(y)/\partial y$ are calculated according to the y value;
6. Coefficients a_0 and a_1 are adjusted according to the above expressions;
7. The procedure is repeated starting from p. 2;
8. The procedure is repeated starting from p. 1, and results are averaged across the set of injection of the initial conditions.

12.7 Two-Layer Neural Network in the Self-Learning Mode

The two-layer neural network with four neurons of two solutions in the first layer and a neuron of $K_p = 5$ solutions in the second layer was considered initially. In this case,

$$y = F(g) = 1 + \frac{1}{2}\sum_{k_p=1}^{K_p}\left[\operatorname{sign}\left\{\sum_{j=1}^{H_1=4} a_j \operatorname{sign}\left(\sum_{i=0}^{N=2} a_{i,j}x_i\right) - k_p\right\} + 1\right]$$

The following expressions obtained using the results of Chap. 9 form the basis for the design of the closed-loop two-layer neural network in the self-learning mode:

$$a_j(n+1) = a_j(n) + K^{*}\left[\mathbf{x}(n) - \mathbf{b}(y,n)\right]^T \frac{\partial \mathbf{b}(y,n)}{\partial y} \overline{\operatorname{sign}\left[\sum_{i=0}^{N} a_{ij}(n)x_i(n)\right]}^{m_n}, \quad j = 1,\ldots,5$$

$$a_{ij}(n+1) = a_{ij}(n) + K^{**}\left[\mathbf{x}(n) - \mathbf{b}(y,n)\right]^T \frac{\partial \mathbf{b}(y,n)}{\partial y} \overline{\operatorname{sign}\left[a_j(n)x_i(n)\right]}^{m_n}, \quad i = 1,\ldots,N;\ j = 1,\ldots,5$$

$$\mathbf{b}(y,n+1) = \mathbf{b}(y,n) + K^{***}\left[\mathbf{x}(n) - \mathbf{b}(y,n)\right]$$

The experimental investigation of this algorithm showed the low convergence rate at the search of some local mode due to the use of the neurons with two solutions that "desensitize" the information about the secondary optimization functional gradient. These neurons were substituted by the neurons with a continuum of solutions. In this case,

$$y = 1 + \frac{1}{2}\sum_{k_p=1}^{K_p=5}\left[\frac{2}{\pi}\operatorname{arctg} B\left\{\sum_{j=1}^{H_1=4} a_j \frac{2}{\pi}\operatorname{arctg} B\left(\sum_{i=0}^{N} a_{ij}x_i\right) - k_p\right\} + 1\right]$$

$$a_j(n+1) = a_j(n) + \overline{K^{*}\left[x - b(y,n)\right]^T \frac{\partial \mathbf{b}(y,n)}{\partial y}} \times \frac{2}{\pi}\sum_{k_p=1}^{K_p} \frac{B}{1 + \sum_{j=1}^{H_1} a_j(n)\frac{2}{\pi}\operatorname{arctg} B\left[\sum_{i=0}^{N} a_{ij}(n)x_i(n) - k_p\right]^2} \overline{\operatorname{arctg} B\left[\sum_{i=0}^{N} a_{ij}(n)x_i(n)\right]}^{m_n} \tag{12.10}$$

$$a_{ij}(n+1) = a_{ij}(n) + \overline{K^{**}\left[x - b(y,n)\right]^T \frac{\partial b(y,n)}{\partial y}} \times \frac{2}{\pi}\sum_{k_p=1}^{K_p} \frac{B^2 a_j(n)x_i(n)}{1 + \sum_{j=1}^{H_1} a_j(n)\frac{2}{\pi}\operatorname{arctg} B\left[\sum_{i=0}^{N} a_{ij}(n)x_i - k_p\right]^2} \times \overline{\frac{1}{1 + \left[B\sum_{i=0}^{N} a_{ij}(n)x_i(n)\right]^2}}^{m_n} \tag{12.11}$$

$$b(y,n+1)=b(y,n)+K^{***}\overline{\left[x-b(y,n)\right]}^{m_n} \tag{12.12}$$

The realized adjustment algorithm consists of the following stages:

1. The random or deterministic initial values of the classes' center coordinates and adjustable coefficients of this two-layer neural network are fed into the memory of the system;
2. The calculation of number of vectors $\partial\mathbf{b}(y)/\partial y$ is performed;
3. Pattern $\mathbf{x}$ enters the neural network input;
4. The y value is calculated according to the obtained $\mathbf{x}$ and multilayer neural network state at the current time instant;
5. The corresponding vectors $\mathbf{b}(y)$ and $\partial\mathbf{b}(y)/\partial y$ are selected according to the y value;
6. New values of the adjustable neural network coefficient class centers are calculated using the results of p. 3,5;
7. When the next pattern enters the input, the algorithm according to p. 4–6 is repeated;
8. The algorithm is repeated according to p. 1–7 after the detection of the local extremum.

Figure 12.36 shows the equal value lines for the distribution density $f(\mathbf{x})$ used in the experimental investigation of this algorithm. The optimal values of the adjustable coefficients for the first layer neurons are $a_{11}=9$; $a_{12}=15$; $a_{13}=21$; $a_{14}=27$; $a_{21}=1$; $a_{22}=1$; $a_{23}=1$; $a_{24}=1$; $a_{31}=1$; $a_{32}=1$; $a_{33}=1$; and $a_{34}=1$.

The neuron of the second layer with K_p solutions must realize a logical function represented in Table 12.6. The elaboration of the correct solution y requires the correct formation of the corresponding intermediate value of the analogous output signal

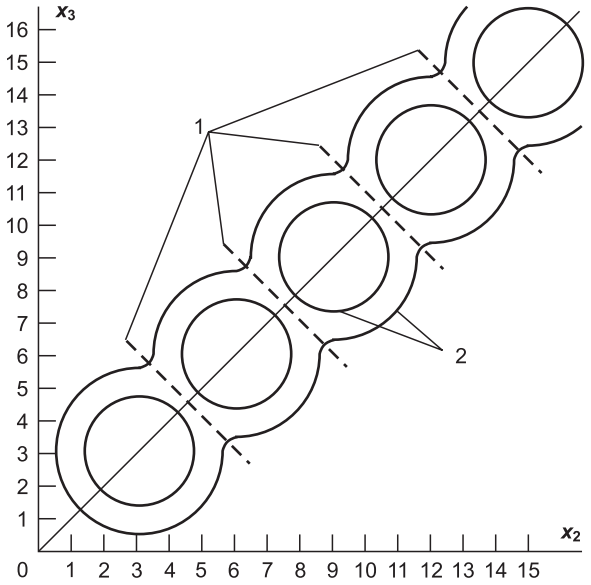


Fig. 12.36.
The equal value lines for the distribution density $f(x)$:
1 – optimal position of the divisional hyperplanes;
2 – equal value lines for the input signal distribution density for the different distribution dispersions representing $f(x)$ modes

Table 12.6.
Logical function to be realized by the neuron of the second layer with K_p solutions

y	1	2	3	4	5
y_1	−1	1	1	1	1
y_2	−1	−1	1	1	1
y_3	−1	−1	−1	1	1
y_4	−1	−1	−1	−1	1

$g(n) = y - 0.5$. The system of algebraic equations for the calculation of the optimal coefficients for the second layer neuron can be obtained on this basis:

$$-a_1 - a_2 - a_3 - a_4 + 2.5 = 0.5$$

$$a_1 - a_2 - a_3 - a_4 + 2.5 = 1.5$$

$$a_1 + a_2 - a_3 - a_4 + 2.5 = 2.5$$

$$a_1 + a_2 + a_3 - a_4 + 2.5 = 3.5$$

$$a_1 + a_2 + a_3 + a_4 + 2.5 = 4.5$$

Consequently,

$$a_1 = a_2 = a_3 = a_4 = 0.5.$$

The experiments with such a two-layer neural network were carried out according to the following plan:

1. Experiments with different distribution dispersions representing $f(x)$ modes (Fig. 12.36);
2. Coefficients of the second layer neuron and the classes' center values are optimal. The following different conditions were taken for the first layer neurons:
   a Neuron coefficients were optimal;
   b Initial values of neuron coefficients were taken with equal deviations from the optimal ones;
   c Initial values of neuron coefficients were taken with different deviations from the optimal ones.
3. The second layer neuron coefficients were optimal; then they were similar to p. 2a, 2b but with the initial classes' center values that were not optimal;
4. The first layer neuron coefficients and classes' center values were optimal. The initial second layer neuron coefficients were not optimal;
5. Similar to p. 3 but with initial classes' center values that were not optimal;
6. The first and second layer neuron coefficients and classes' center values were not optimal;
7. All the aforementioned experiments were performed under different but deterministic initial conditions. The final experiment was based on the random initial conditions for neuron coefficients and the classes' centers.

Experiment p. 1a showed the stability of the two-layer neural network in the global extremum of a special average risk function. Figure 12.37 shows the results for the case when the initial neuron coefficients were optimal in both layers but the classes' centers were not optimal. Figure 12.38 shows the results for the case when the first layer initial

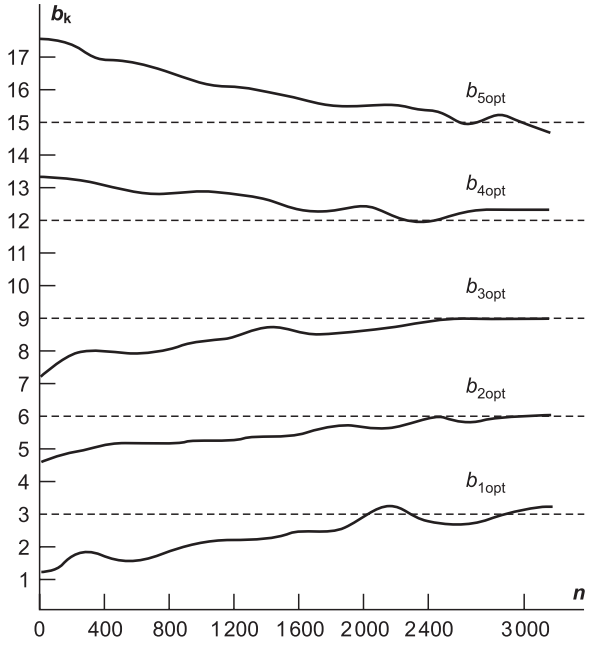


Fig. 12.37.
Dynamics of adjustment by the classes' center coordinates

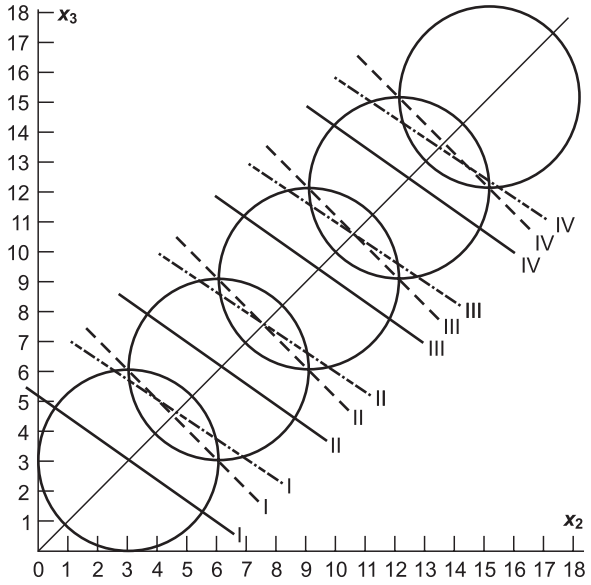


Fig. 12.38.
Investigation results for the two-layer neural network in the self-learning mode: —— initial hyperplane position; – – – – – optimal hyperplane position; – – · – – hyperplane position after 3000 iterations

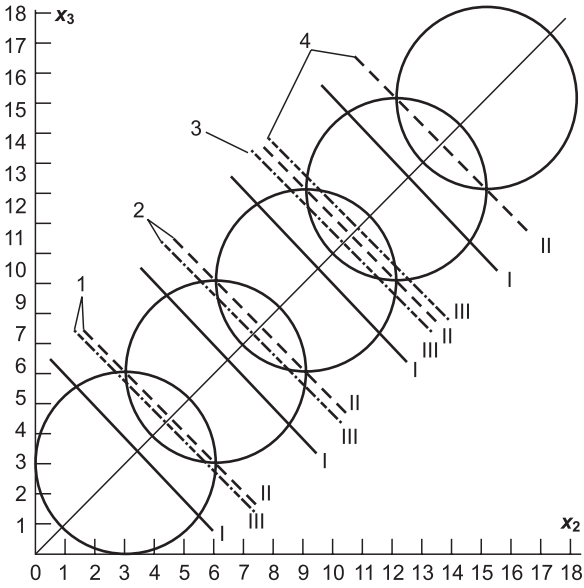


Fig. 12.39.
Investigation results for the two-layer neural network in the self-learning mode: *I* – initial hyperplane position; *II* – optimal hyperplane position; *III* – hyperplane position after 3 000 iterations

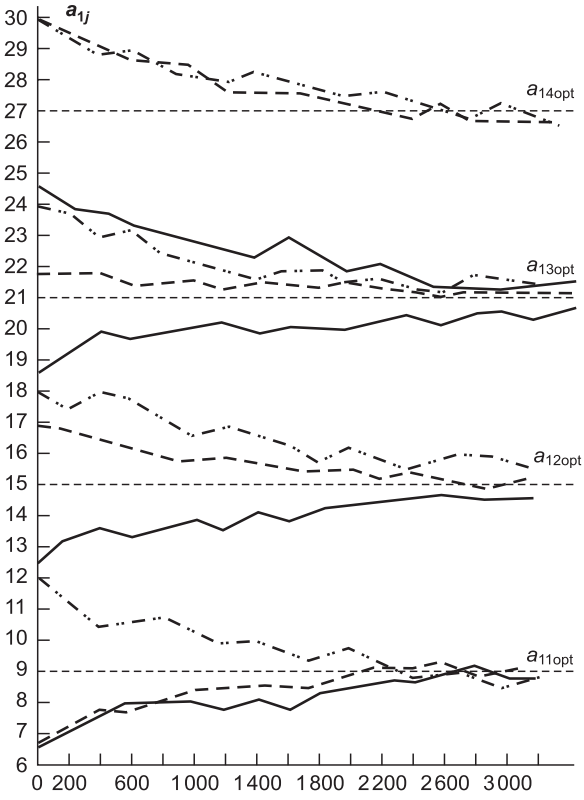


Fig. 12.40.
Investigation results for the two-layer neural network in the self-learning mode: *1–4* – final positions of hyperplanes realized by neurons 1–4 of the first layer, respectively

neuron coefficients were not optimal but the second initial neuron coefficients and classes' centers were optimal. It is seen that the selected location of the input signal distribution density modes makes the hyperplanes (*I, II, III, IV*) insensitive to the rotation. The quality functional is almost constant here, and this fact is confirmed experimentally. It is therefore advisable to investigate only the dynamics of the threshold a_{ij} adjustment ($j = 1, 2, 3, 4$) under the given fixed $f(\mathbf{x})$ mode location and, as a rule, not to investigate the dynamics of the hyperplane coefficients.

Figure 12.39 shows the results for the case when the initial coefficients of the second layer neuron were optimal but the initial values of first layer neuron coefficients and the classes' centers were taken with equal negative deviations from their optimal values. The threshold adjustment dynamics is shown in Fig. 12.40 (*solid line*). The case of positive deviations is shown by the *dot-and-dash line*. Figure 12.40 (*dashed line*) and 12.41 show the experimental results for the case when the initial coefficients of the second layer neuron were optimal but the initial values of first layer neuron coefficients and the classes' centers were taken with different centers by the sign and magnitude deviations from their optimal values.

The goal of the experiment p. 6 was to investigate the influence of the distribution dispersion upon the recognition quality. The initial conditions in this experimental series were the same, whereas the dispersion σ^2 was different (Fig. 12.42).

These experiments show that the self-learning problem can be solved for σ^2 not more than $\sigma^2_{\max} = 1.5$. This is reasonable because it is impossible to select locally concentrated objects under the large σ^2 (classes with a large intersection). Hence, the self-learning methods based on such a selection appear to be invalid under such conditions.

Figure 12.43 shows the experimental results for the case when the initial coefficients of the first layer neuron classes' centers were optimal but the initial values of the second layer neuron coefficients were not optimal.

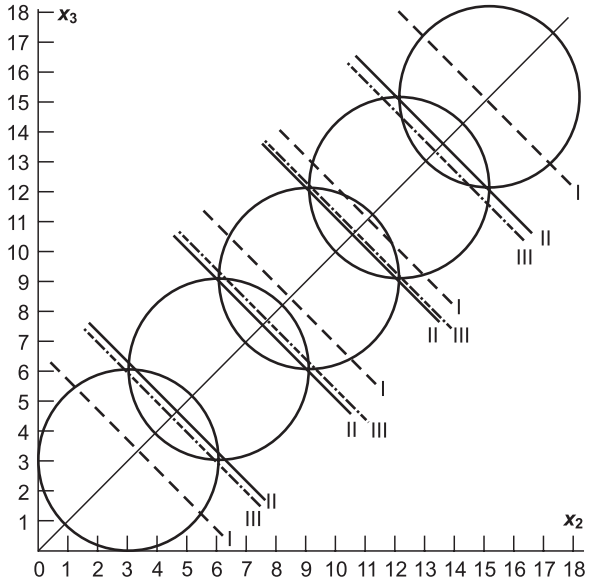


Fig. 12.41.
Investigation results for the two-layer neural network in the self-learning mode: *I* – initial hyperplane position; *II* – optimal hyperplane position; *III* – hyperplane position after 3000 iterations

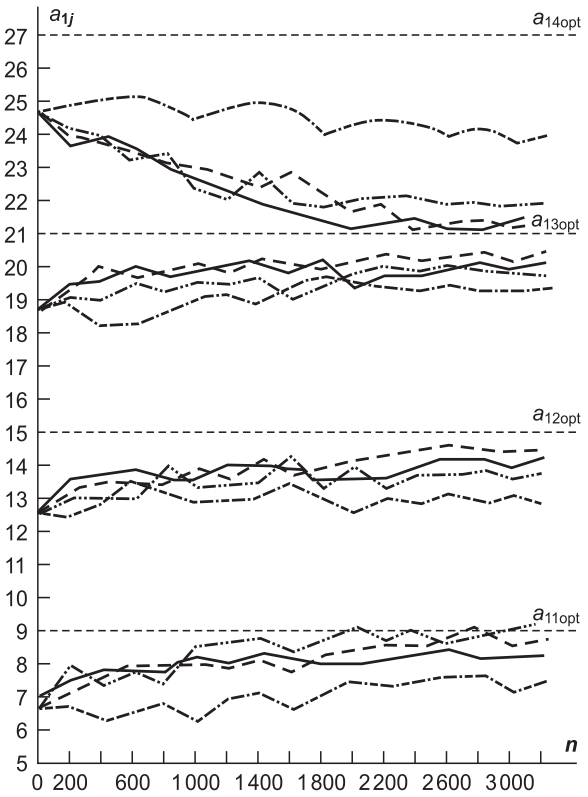


Fig. 12.42.
Investigation results for the two-layer neural network in the self-learning mode: – – – – $\sigma^2 = 1$; —— $\sigma^2 = 1.5$; —··— $\sigma^2 = 2$; – – · – – $\sigma^2 = 2.5$

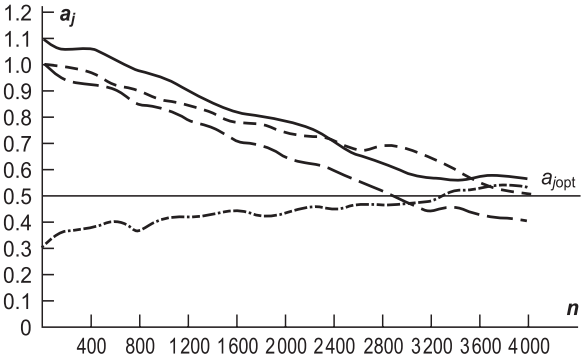


Fig. 12.43.
Investigation results for the two-layer neural network in the self-learning mode: – – · – – a_1; – – – – a_2; — — a_3; —— a_4

The adjustment procedure termination criterion in this experiment was the location of the curves inside the "tube" with a diameter of 0.2 (Fig. 12.43) and a length of 5 000 iterations. Figure 12.44 illustrates this experiment.

Figure 12.45 shows the experimental results with another distribution density form as compared with the typical density form $f_x(\mathbf{x})$.

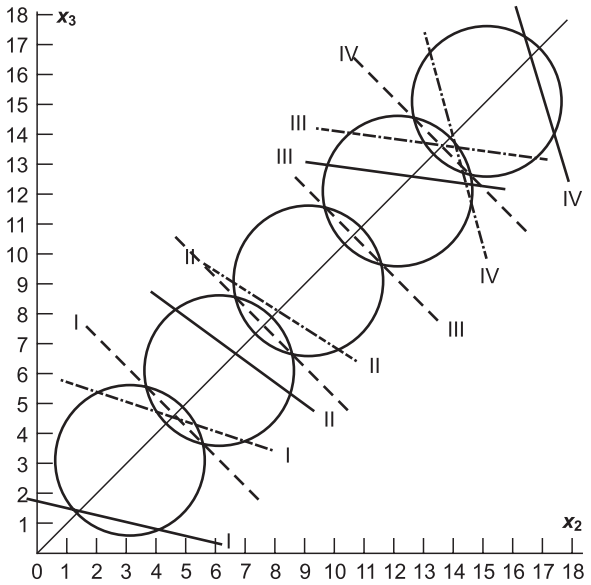


Fig. 12.44.
Investigation results for the two-layer neural network in the self-learning mode: *I* , *II* , *III* , *IV* – numbers of the respective neurons of the first layer; – – – – optimal hyperplane position; —— initial hyperplane position; —·—·— hyperplane position after 5 000 iterations

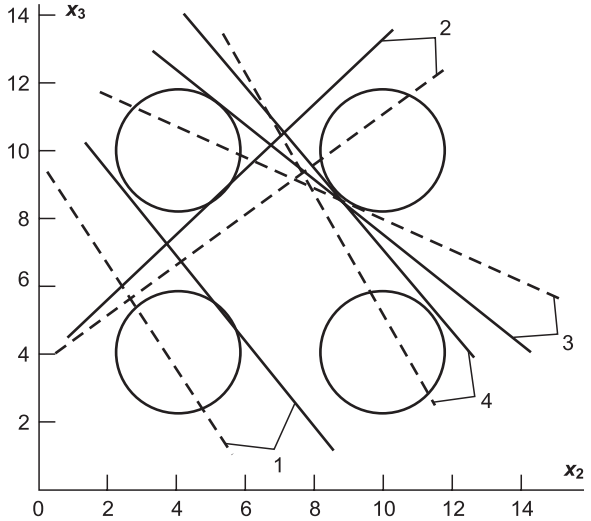


Fig. 12.45.
Investigation results for the two-layer neural network in the self-learning mode: *1–4* – numbers of the respective neurons of the first layer

12.8
About Some Engineering Methods for the Selection of Matrix Parameters in the Multilayer Neural Network Closed Cycle Adjustment Algorithms

It is unlikely to have enough information for making matrix K^* non-diagonal if only the first derivative of the optimization functional is estimated in the process of the closed-cycle adjustment algorithm design. This matrix in the simplest case is a unit matrix multiplied by some constant or time-dependent coefficient. However, as the

above experiments have shown, there are some reasons to make this coefficient different for the adjustment of different layers of the multilayer neural network. As it was mentioned in Chap. 9, the main goal of the use of stochastic approximation methods is to provide the zero random and dynamic errors in determination of the adjustment coefficient vector in the stationary state. But the use of such methods results in the increase of corresponding dynamic errors in the transient state, i.e., in the adjustment mode. It is unlikely that it is necessary to provide the zero random error in the stationary state during the multilayer neural network adjustment. Some finite adjustment coefficient distribution dispersion is admissible due to the relatively smooth properties of the secondary optimization functional in its extremum point. This finite dispersion of distribution $f_a(\mathbf{a})$ does not result in the significant increase of the secondary optimization functional, and it can be provided by the time-invariant matrix K^*. Two engineering methods are possible in this case for the selection of the matrix K^* coefficients. The first one is based on the analysis of the a priori given problem complexity determined by the modality $f_x(\mathbf{x})$ under the fixed dimensions of the feature space.

The second method based also on the analysis of experiments shows that the objective necessity to estimate the secondary optimization functional emerges in the real situation during the multilayer neural network adjustment process. If the dependence of the secondary optimization functional upon n strongly oscillates, then it is necessary to decrease K^*. If this dependence is sufficiently smooth, then it is necessary to increase K^* in order to decrease systematic adjustment error up to the emergence of oscillations. The first method for selection of K^* can be used for selection of the initial value of K^* in the second method.

12.9 Design of the Multilayer Neural Network for the Matrix Inversion Problem

Let us consider the design of the multilayer neural network and its closed-cycle adjustment algorithm for matrix inversion when its dimensionality is 2×2. Since the result also represents some matrix 2×2 then the multilayer neural network output must consist of four neurons with a continuum of solutions. The minimum variant of the open-loop three-layer neural network structure is shown in Fig. 12.46 in the form of a graph-scheme.

The initial conditions on the adjustment coefficients of the first layer neurons must be selected in such a way that under $B=\infty$, four hyperplanes must divide the initial

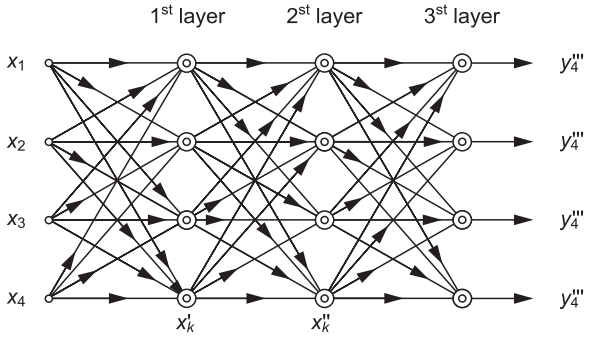


Fig. 12.46. Graphs of the multilayer neural network for the matrix inversion

feature space into the regions with equal hyper-volumes. The initial condition selection for the second and third layers must be performed in a similar way as in the case of the first layer because the neurons with a continuum of solutions are also used here.

The formation of the learning sample for the considered multilayer system is carried out according to the following expressions:

$$\mathbf{X}=\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} \; ; \quad \mathbf{X}^{-1}=\begin{bmatrix} \frac{x_4}{D} & -\frac{x_2}{D} \\ -\frac{x_3}{D} & \frac{x_1}{D} \end{bmatrix} \; ; \quad D=x_1x_4-x_2x_3$$

There are practically no constraints upon the amplitude of the multilayer neural network input signal. But the amplitude of the output signal is limited by the interval [–1, + 1] due to the output neurons' specific character. This requires some normalization of the input signal in such a way that all the output signal components do not exceed the interval [–1, + 1]. Such normalization must be performed in the following way. Let **X** be the initial matrix and

$$x=\max_{i=1,2,3,4}\left\{\left[x_i\right]\right\}$$

Division of **X** by x gives the matrix $\mathbf{X}^{-1}$ with elements belonging to the interval [–1, + 1]. Let us designate

$$D_1=\begin{vmatrix} \frac{x_1}{x} & \frac{x_2}{x} \\ \frac{x_3}{x} & \frac{x_4}{x} \end{vmatrix}$$

Then

$$\mathbf{X}^{-1}=\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix}^{-1}=\left(xD_1\begin{bmatrix} \frac{x_1}{xD_1} & \frac{x_2}{xD_1} \\ \frac{x_3}{xD_1} & \frac{x_4}{xD_1} \end{bmatrix}\right)^{-1}=\frac{1}{xD_1}\left(D_1\begin{bmatrix} \frac{x_1}{x} & \frac{x_2}{x} \\ \frac{x_3}{x} & \frac{x_4}{x} \end{bmatrix}^{-1}\right)=\frac{1}{xD_1}\begin{bmatrix} \frac{x_4}{x} & -\frac{x_2}{x} \\ -\frac{x_3}{x} & \frac{x_1}{x} \end{bmatrix}$$

Consequently, the multiplication of matrix **X** elements at the input by $1/xD_1$ and the following application of this matrix to the inversion system provide the matrix

$$\begin{bmatrix} x_4 & -x_2 \\ -x_3 & x_1 \end{bmatrix}\frac{1}{x}$$

with elements inside the interval [–1, + 1]. Multiplication of this matrix by $1/xD_1$ gives the final result $\mathbf{X}^{-1}$.

The structure of the open-loop multilayer neural network is described by the following relationships:

$$y'_{h_1} = \frac{2}{\pi}\operatorname{arctg} B\left(\sum_{h_0=1}^{4} a_{h_1 h_0} x_{h_0} + a_{0h_1}\right), \quad h_1 = 1,2,3,4$$

$$y''_{h_2} = \frac{2}{\pi}\operatorname{arctg} B\left(\sum_{h_1=1}^{4} a_{h_2 h_1} x_{h_0} + a_{0h_2}\right), \quad h_2 = 1,2,3,4$$

$$y'''_{h_3} = \frac{2}{\pi}\operatorname{arctg} B\left(\sum_{h_2=1}^{4} a_{h_3 h_2} x_{h_0} + a_{0h_3}\right), \quad h_3 = 1,2,3,4$$

The teacher instruction ε_{h_3} for the multilayer system must be formed in the algorithmic way using one of the known matrix inversion algorithms and inversion precision control. The expressions for the mean-root-square error of the elements of the matrix

$$\hat{a}_{2g} = \overline{x_g^2}^{m_n} = \frac{1}{4}\overline{\sum_{h_3=1}^{4}(y'''_{h_3} - \varepsilon_{h_3})^2}^{m_n} = \frac{1}{4}\overline{\sum_{h_3=1}^{4}(x_{gh_3})^2}^{m_n}$$

have the following form:

$$\frac{\overline{\partial x_g^2}^{m_n}}{\partial a_{h_1 h_0}} = \frac{4B^3}{\pi^3}\overline{\frac{x_{h_0}}{1+(g_{h_1}^1)^2}\sum_{h_3=1}^{4}\frac{x_{gh_3}}{1+(g_{h_3}^3)^2}\sum_{h_2=1}^{4}\frac{a_{h_3h_2}a_{h_2h_1}}{1+(g_{h_2}^2)^2}}^{m_n}$$

$$\frac{\overline{\partial x_g^2}^{m_n}}{\partial a_{h_2 h_1}} = \frac{2B^2}{\pi^2}\overline{\frac{y'_{h1}}{1+(g_{h_2}^2)^2}\sum_{h_3=1}^{4}\frac{x_{gh_3}a_{h_3h_2}}{1+(g_{h_3}^3)^2}}^{m_n}$$

$$\frac{\overline{\partial x_g^2}^{m_n}}{\partial a_{h_3 h_2}} = \frac{B}{\pi}\overline{\frac{y''_{2h_2}x_{gh_3}}{1+(g_{h_3}^3)^2}}^{m_n}$$

Here

$$g_{h_1}^1 = \sum_{h_0=1}^{4} a_{h_1 h_0} x_{h_0} + a_{0h_1}$$

$$g_{h_2}^2 = \sum_{h_1=1}^{4} a_{h_2 h_1} y'_{h_1 1} + a_{0h_2}$$

$$g_{h_3}^3 = \sum_{h_2=1}^{4} a_{h_3 h_2 1} y'_{h_2 2} + a_{0h_3}$$

These expressions form the basis for the design of the adaptation algorithm of the multilayer system aimed at the inversion of matrix 2×2.

12.10 Design of the Multilayer Neural Network for the Number Transformation from the Binary System into the Decimal One

The transformation of the four-bit binary number will be considered as an example. The system must realize the desired relationship "input-output" described by the multiple-valued logic function $\varepsilon(\mathbf{x})$ represented in Table 12.7 after the termination of the closed-cycle adjustment mode.

Table 12.7 provides the formation of the learning sample at the system input along with the teacher instruction ε by selection of the table columns.

The open-loop neural network is described in this case by the following expression:

$$x_3 = 1 + \frac{1}{2}\sum_{k_p=1}^{K_p-1}\left[\operatorname{sign}\left(\sum_{h_2=1}^{3} a_{h_3h_2}\frac{2}{\pi}\operatorname{arctg} B\sum_{h_1=1}^{4} a_{h_2h_1}\frac{2}{\pi}\operatorname{arctg} B\sum_{h_0=0}^{4} a_{h_1h_0}x_{h_0} - a_{k_p,k_{p+1}}\right)+1\right]$$

$$a_{k_p,k_{p+1}} = k_p - 1; \quad K_p = 10$$

Consequently,

$$\operatorname{sign}\frac{\partial}{\partial a_{h_3h_2}}x_3 = \operatorname{sign}\left[\sum_{h_1=1}^{4} a_{h_2h_1}\sum_{h_0=0}^{4} a_{h_1h_0}x_{h_0}\right]$$

$$\operatorname{sign}\frac{\partial}{\partial a_{h_2h_1}}x_3 = \operatorname{sign}\left[a_{h_3h_2}\sum_{h_0=0}^{4} a_{h_1h_2}x_{h_0}\right]$$

$$\operatorname{sign}\frac{\partial}{\partial a_{h_1h_0}}x_3 = \operatorname{sign}\left[a_{h_3h_2}a_{h_2h_1}x_{h_0}\right]$$

These expressions form the basis for the design of the adjustment algorithm of the multilayer system aimed at the number transformation from the binary system into the decimal one.

Table 12.7. Formation of the learning sample

ε		0	1	2	3	4	5	6	7	8	9
$\mathbf{x}_{h_0}$	$x_{h_0}^1$	−1	−1	−1	−1	−1	−1	−1	−1	1	1
	$x_{h_0}^2$	−1	−1	−1	−1	1	1	1	1	−1	−1
	$x_{h_0}^3$	−1	−1	1	1	−1	−1	1	1	−1	−1
	$x_{h_0}^4$	−1	1	−1	1	−1	1	−1	−1	−1	1

12.11 Investigation of the Multilayer Neural Network under the Arbitrary Teacher Qualification

The design of the optimal neural network model in the case of the arbitrary objective and subjective teacher qualification was described in Chap. 5. We considered below the case $K = 2$ and arbitrary objective teacher qualification b_0.

The pattern recognition system represented a two-layer neural network based on the neurons with arc tangent characteristics and $B = 5$. The adjustment algorithm was modeled in the learning ($b_c = 1$) and self-learning ($b_c = 0$) modes. The algorithm block diagram is represented in Fig. 12.47.

The main aim of the experimental study was to test the system operation capability. The plan of experiments included two main points:

1. The investigation of the system behavior under the optimal coefficient values and different relationships between b_0 and b_c;
2. The investigation of the system dynamics for different b_0 and b_c and non-optimal neurons.

The pseudo-random-number generator with the distribution close to the normal one and with equal covariance matrices for both classes was used as the generator of the input signals. The experimental study showed the following results concerning p. 1:

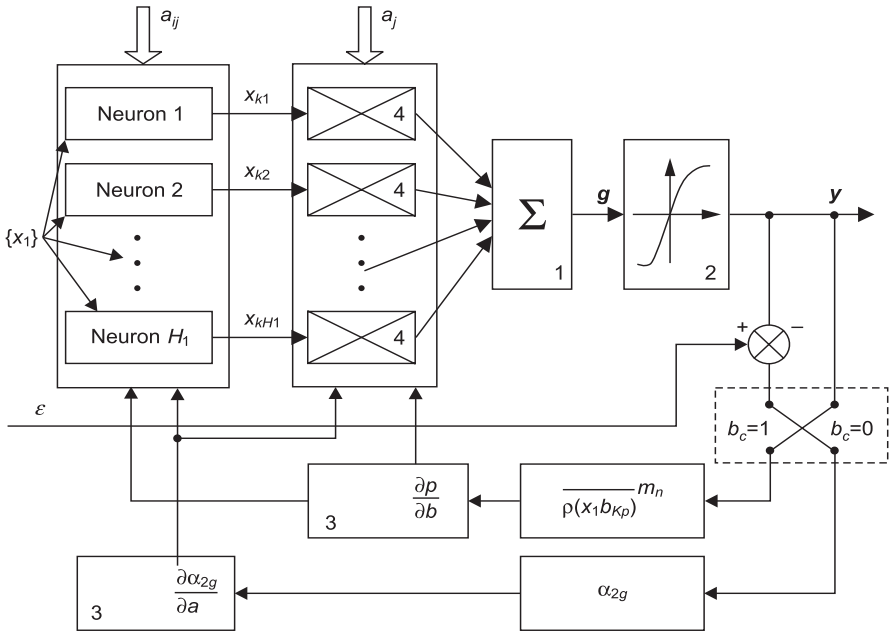


Fig. 12.47. Block diagram of the neural network with subjective teacher qualification: *1* – adder unit; *2* – nonlinear conversion device; *3* – units of gradient calculations; *4* – multiplier unit

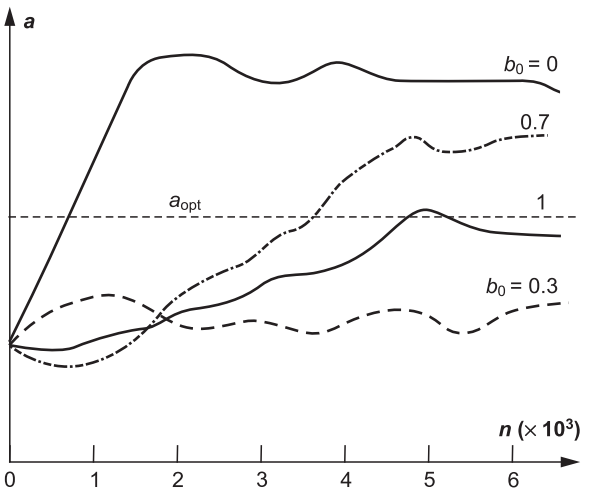


Fig. 12.48.
Dynamics of the system coefficient changes under $b_c = 1$ and different b_0

1. The oscillations of the system coefficients around their optimal values are observed in the case $b_0 = b_c$;
2. The gradual detuning of the system takes place in the case $b_c = 1$. The detuning increases as b_0 changes from 1 to 0;
3. The system remains in the optimal state in the case $b_c = 0$ independently of b_0.

The investigations under the non-optimal initial coefficients showed that in the case $b_0 = b_c$, the adjustment procedure leads the system into the optimal state. In the case $b_c = 1$ and $b_c \neq b_0$, the system is not adjustable in spite of the long adjustment duration (Fig. 12.48).

12.12 Analytical Methods of Investigations of the Neural Network Closed Cycle Adjustment

We describe below the general analytical methods for the investigation of the closed-cycle neural network adjustment. The particular examples are used for illustration. The complications of the described methods are discussed.

The general methods of analysis of the closed-cycle neural network adjustment are similar by its structure to the methods used for the analysis of the open-cycle adjustment. They include the following stages:

1. The analysis of the probability distribution density for the vector of the secondary optimization functional gradient estimation;
2. The derivation of the stochastic differential equation for the change of the adjustable coefficient distribution density in the course of the adjustment procedure;
3. The solution of this equation;
4. The determination of the probability distribution for the correct recognition by means of integrating across the feature space and across the neural network state space (the space of the adjustment coefficients).

In principle, the problem of the parameter matrix K^* selection must be performed according to the results of p. 3. However, it will be shown below that this task is rather complicated. In this section, we consider a linear threshold element with the minimum magnitude of the first discrete error moment optimization criterion.

The recurrent relationship for the neuron with $|\alpha_{1g}|$ minimization in the case $N = m_n = 1$ was obtained in the following form:

$$a_0(n+1) = a_0(n) - K^* x_g(n)$$

The first stage of analysis. This is the problem of the random walk across the one-dimensional grid. Such a walk is described by the Markovian chain with an infinite number of states. The probabilities of transition from the state mK^* to the state $(m+1)K^*$, $(m-1)K^*$ and mK^* are

$$P\left[mK^* \middle| (m+1)K^*\right] = \frac{1}{2}\left[1 - \Phi_1\left(mK^*\right)\right]$$

$$P\left[mK^* \middle| (m-1)K^*\right] = \frac{1}{2}\Phi_2\left(mK^*\right)$$

$$P\left[mK^* \middle| mK^*\right] = \frac{1}{2}\left[1 + \Phi_1\left(mK^*\right) - \Phi_2\left(mK^*\right)\right]$$

Here Φ is the cumulative detection probability.

The second stage of analysis. The stochastic difference equation for the change of the probability distribution density of the threshold a_0 has the following form:

$$W_{n+1}\left(mK^*\right) = W_n\left[(m-1)K^*\right]\frac{1}{2}\left\{1 - \Phi_1\left[(m-1)K^*\right]\right\}$$
$$+ W_n\left(mK^*\right)\frac{1}{2}\left[\Phi_1\left(mK^*\right) + 1 - \Phi_2\left(mK^*\right)\right] + W_n\left[(m+1)K^*\right]\frac{1}{2}\Phi_2\left[(m+1)K^*\right]$$

The third stage of analysis. The solution of this stochastic difference equation is rather difficult. Let us consider the stationary state ($n = \infty$).

Taking $a_0(0) = 0$ and proceeding to the limit $n \to \infty$ one obtains

$$W\left[(m-1)K^*\right]\frac{1}{2}\left\{1 - \Phi_1\left[(m-1)K^*\right]\right\} + W\left[(m+1)K^*\right]\frac{1}{2}\Phi_2\left[(m+1)K^*\right]$$
$$- W\left(mK^*\right)\frac{1}{2}\left[1 - \Phi_1\left(mK^*\right) + \Phi_2\left(mK^*\right)\right] = 0$$

Consequently,

$$W\left[(m-1)K^*\right]\frac{1}{2}\left\{1 - \Phi_1\left[(m-1)K^*\right]\right\} - W\left(mK^*\right)\frac{1}{2}\Phi_2\left(mK^*\right)$$
$$= W\left(mK^*\right)\frac{1}{2}\left[1 - \Phi_1\left(mK^*\right)\right] - W\left[(m+1)K^*\right]\frac{1}{2}\Phi_2\left[(m+1)K^*\right] = C$$

It follows from the normalization condition for the distribution density of coefficient a_0 by m that $W(mK^*) = 0$. Consequently, $C = 0$ and

$$W\left[(m-1)K^*\right]\frac{1}{2}\left\{1-\Phi_1\left[(m-1)K^*\right]\right\} = W\left(mK^*\right)\frac{1}{2}\Phi_2\left(mK^*\right)$$

Taking $W(0) = A$, one obtains

$$W\left(K^*\right) = A\frac{1-\Phi_1(0)}{\Phi_2\left(K^*\right)};\quad W\left(2K^*\right) = W(\Delta)\frac{1-\Phi_1\left(K^*\right)}{\Phi_2\left(2K^*\right)}$$

$$W\left(-K^*\right) = A\frac{\Phi_2(0)}{1-\Phi_1\left(K^*\right)}$$

$$W\left(-2K^*\right) = W\left(-K^*\right)\frac{\Phi_2\left(-K^*\right)}{1-\Phi_1\left(-2K^*\right)}$$

In the general case,

$$\begin{aligned} W\left(mK^*\right) &= A\prod_{k=1}^{m-1}\frac{1-\Phi_1\left[(k-1)K^*\right]}{\Phi_2\left[kK^*\right]} \\ W\left(-mK^*\right) &= A\prod_{k=1}^{-m}\frac{\Phi_2\left[(k+1)K^*\right]}{\Phi_2\left[kK^*\right]} \end{aligned} \tag{12.13}$$

The value $A = W(0)$ is determined from the normalization condition of the density W by m. Function $W(.)$ represents the distribution density for the adjusted coefficient a_0 in the stationary state. Function $\frac{1}{2}[1 - \Phi_1(mK^*)]$ is a steadily decreasing function from $\frac{1}{2}$ to 0 in the interval $-\infty < m\Delta < +\infty$. Function $\frac{1}{2}\Phi_2(mK^*)$ is a steadily increasing function from 0 to $\frac{1}{2}$ in the interval $-\infty < m\Delta < +\infty$. Function $\frac{1}{2}[1 - \Phi_1(mK^*)] + \Phi_2(mK^*)$ has its maximum at the root of equation

$$1 - \Phi_1[(mK^*)] = \Phi_2(mK^*) \tag{12.14}$$

Let the root of Eq. (12.14) be $mK^* = \widehat{\theta}$. Then at $m\Delta \geq \widehat{\theta}$,

$1-\Phi_1\left(mK^*\right) \leq \Phi_2\left(mK^*\right) < \Phi_2\left[(m+1)K^*\right]$, i.e.

$$\frac{\Phi_2\left(mK^*\right)}{1-\Phi_1\left[(m-1)K^*\right]} > 1$$

Respectively, at $mK^* \le \hat{\theta}$ one obtains

$$\Phi_2\left(mK^*\right) \le 1-\Phi_1\left(mK^*\right) < 1-\Phi_1\left[(m-1)K^*\right]$$

$$\frac{\Phi_2\left(mK^*\right)}{1-\Phi_1\left[(m-1)K^*\right]} < 1$$

Therefore, if $\hat{\theta}/K^*$ is an integer number, then

$$W\left(\hat{\theta}-\Delta\right) = W\left(\hat{\theta}\right)\frac{\Phi_2\left(\hat{\theta}\right)}{1-\Phi_1\left(\hat{\theta}-\Delta\right)} < W\left(\hat{\theta}\right)$$

$$W\left(\hat{\theta}+\Delta\right) = W\left(\hat{\theta}\right)\frac{1-\Phi_1\left(\hat{\theta}\right)}{\Phi_2\left(\hat{\theta}+\Delta\right)} < W\left(\hat{\theta}\right)$$

Consequently, $\hat{\theta}$ is the mode of distribution of the threshold value as a random variable, and it provides the equality of the conditional risk functions for the pattern assemblages of the first and second classes.

It follows from (12.13) that the mathematical expectation and distribution dispersion of the threshold are finite.

For the neuron with arbitrary memory m_n in the adjustment block ($m_n = \text{const}, N = 1$),

$$a_0(n+1) = \begin{cases} a_0(n) \quad , & \text{if} \quad n+1 \neq im_n, \quad i = 1,2,3,\ldots \\ a_0(n) + \dfrac{K_n^*}{m_n} \displaystyle\sum_{l=(i-1)m_n+1}^{im_n} \left\{\varepsilon(l) - \operatorname{sign}\left[x(l) - a_0(l)\right]\right\} \quad , & \text{if} \quad n = im_n \end{cases} \tag{12.15}$$

The expression (12.15) is valid in spite of the remark made in Chap. 9 about the impossibility to design the analytical adjustment algorithm with arbitrary values of m_n in the general case with the minimum $|\alpha_{1g}|$ criterion and the system with two solutions. This is explained by the fact that the expression (12.15) concerns the particular one–dimensional case ($N = 1$) with $x_0 = -1 = \text{const}$. The coefficient adjustment in (12.15) is performed after each m_n cycle of patterns entering the system input.

The expression for the transfer probability in the given Markovian chain is similar to the case $m_n = 1$:

$$P\left[x_g(n) = -2\right] = \frac{1}{2}\left[1-\Phi_1\left(mK^*\right)\right]$$

$$P\left[x_g(n) = 2\right] = \frac{1}{2}\Phi_2\left(mK^*\right)$$

$$P\left[x_g(n) = 0\right] = \frac{1}{2}\left[1-\Phi_2\left(mK^*\right)+\Phi_1\left(mK^*\right)\right]$$

where mK^* is the current value of the adjustment coefficient a_0. The value

$$\frac{K_n^*}{m_a}x_g'' = x_g'\left(n = im_n\right) = \frac{K_n^*}{m_n}\sum_{l=(i-1)m_n+1}^{im_n}\left\{\varepsilon(l) - \operatorname{sign}\left[x(l) - a_0(l)\right]\right\}$$

amounts to

$$-K_n^*, -K_n + \frac{K_n^*}{m_n}, \ldots, 0, \ldots, K_n^* - \frac{K_n^*}{m_n}, K_n^*$$

This is the case of the polynomial distribution

$$P\left[x_g''(n = im_n) = 2(l-t)\right] = P_{m_n}\left[l, t, (m_n - l - t)\right] = \frac{m_n!}{l!t!(m_n - l - t)}\left[\frac{1}{2}\Phi_2\left(mK^*\right)\right]^l$$

$$\times\left\{\frac{1}{2}\left[1 - \Phi_1\left(mK^*\right)\right]\right\}^l \left\{\frac{1}{2}\left[1 - \Phi_2\left(mK^*\right) + \Phi_1\left(mK^*\right)\right]\right\}^{m_n - l - t}$$

Here $l, t, m_n - l - t$ are respectively the number of times when $+1, -1, 0$ emerge in x_g. After the change of variables $\xi = l - t$, one can obtain the constraints upon the variable range in the following form:

$$\xi \geq 0, \text{ if } \frac{m_n - \xi}{2} \geq t \geq 0 \;\; ; \;\; \xi < 0, \text{ if } \frac{m_n - \xi}{2} \geq t \geq -\xi$$

The expression for the transition probability in the case $\xi > 0$ has the following form:

$$P\left[m_n K^* \,\middle|\, (m_n + \xi)K^*\right] = \sum_{t=0}^{\left\lfloor \frac{m_n - \xi}{2} \right\rfloor} \frac{m_n!}{(t+\xi)!t!(m_m - 2t - \xi)!} 2^{-m_n}\left[1 - \Phi_1\left(mK^*\right)\right]^t$$

$$\times \Phi_2\left(m_n K^*\right)^{t+\xi}\left[1 + \Phi_1\left(m_n K^*\right) - \Phi_2\left(m_n K^*\right)\right]^{m_n - 2t + \xi}$$

The expression for the transition probability in the case $\xi < 0$ has the same form with substitution of the lower limit by $(-\xi)$. The expression for the transition probability will be unified when the lower limit is taken as $\{\max[0, -\xi]\}$.

The stochastic difference equation for the distribution density of the adjusted coefficient a_0 corresponding to the second stage of analysis has the form

$$W_{lm_n}\left(mK^*\right) = \sum_{k=-m_n}^{m_n} W_{(i-1)m_n}\left[(m-k)K^*\right] P\left[(m-k)K^* \,\middle|\, mK^*\right]$$

where $P[\,]$ is determined by the above expression for the transition probability.

The recurrent relationship for the closed-loop system with $m_n = 1$ in the multidimensional case can be written in the form

$$\mathbf{a}(n+1) = \mathbf{a}(n) + K^* x_g(n)\operatorname{sign}\mathbf{x}(n)$$

This is the case of the walk problem across the $(N+1)$-dimensional grid. Such a walk is described by the multidimensional Markovian chain. The analysis of the closed-loop system here consists in the derivation of expressions for the transition probabilities, derivation of the stochastic equation, and investigation of its solution.

The solution of these problems is rather complicated even in the relatively simple case under consideration.

Neuron with the solution continuum and continuum of pattern classes. Let us consider the case $N = m_n = 1$. Using the criterion of minimization of the discrete error second moment α_{2g}, one obtains

$$a_0(n+1) = a_0(n) - 2K^* \left\{ \varepsilon(n) - F\left[x(n) - a_0(n)\right] \frac{\mathrm{d}F(g)}{\mathrm{d}g} \right\} \tag{12.16}$$

Here

$$x_g(n) = \varepsilon(n) - F\left[x(n) - a_0(n)\right] \;; \quad \frac{\mathrm{d}F(g)}{\mathrm{d}g} = \varphi(g)$$

Let us introduce the following random variables: $A_0(n), Z[n], L[n], X[n], E[n], G[n]$, and $Y[n]$. Their possible values are $a_0[n], z[n], l[n], x[n], \varepsilon[n], g[n]$, and $y[n]$.

The value $G[n]$ is the function of the random variables $A_0[n]$ and $X[n]$:

$$G[n] = X[n] - A_0[n]$$

and $Y[n]$ is the function of the random variable $G[n]$:

$$Y[n] = \varphi(G[n])$$

The value $Z[n]$ is determined in the following way:

$$Z[n] = \left\{E[n] - F(G[n])\right\}\varphi(G[n])$$

$$L[n] = E[n] - F(G[n]); \;\; l[n] = x_g[n]$$

The distribution density of $A_0[n+1]$ will be found in the following form:

$$f\left[a_0(n+1)\right] = \int_{-\infty}^{\infty} f\left[a_0(n+1), x(n)\right]\mathrm{d}x(n) = \int_{-\infty}^{\infty} f\left[a_0(n+1)/x(n)\right] f[x(n)]\,\mathrm{d}x(n) \tag{12.17}$$

In order to determine $f[a_0(n+1)/x(n)]$, one needs to determine $\Phi[z(n)/a_0(n), x(n)]$:

$$\Phi\left[z(n)/a_0(n), x(n)\right] = \int_{x_g(n)} \int_{y(n)<z(n)} f\left[x_g(n), y(n)/a_0(n), x(n)\right]\mathrm{d}x_g(n)\mathrm{d}y(n)$$

$$f_1\left[x_g(n), y(n)/x(n), a_0(n)\right] = f_2\left[x_g(n)/y(n), x(n), a_0(n)\right] f_3\left[y(n)/x(n), a_0(n)\right]$$

$$f_3\left[y(n)/x(n), a_0(n)\right] = \delta\left\{y(n) - \varphi\left[x(n) - a_0(n)\right]\right\}$$

$$f_2\left[x_g(n)/y(n), x(n), a_0(n)\right] = f_2\left[x_g(n)/x(n), a_0(n)\right]$$

because the random value φ is the certain function of the random values of X and A_0:

$$f_2\left[x_g(n)/x(n),a_0(n)\right]=f_4\left\{x_g(n)+F\left[x(n)-a_0(n)\right]/x(n)\right\}$$

$$f_4\left[\varepsilon(n)/x(n),a_0(n)\right]=f_4\left[\varepsilon(n)/x(n)\right]$$

Consequently,

$$f_2\left[x_g(n)/x(n),a_0(n)\right]=f_4^*\left\{x_g(n)+F\left[x(n)-a_0(n)\right]/x(n)\right\}$$

where f_4^* is a new function with fixed $a_0(n)$.

As a result, one obtains

$$f_1\left[x_g(n),y(n)/x(n),a_0(n)\right]=f_4^*\left\{x_g(n)+F\left[x(n)-a_0(n)\right]/x(n)\right\}\delta\left[y(n)-\varphi(g(n))\right]$$

Let us define

$$\Phi\left[z(n)/a_0(n),x(n)\right]=\int\limits_{x_g(n)}\int\limits_{y(n)<z(n)} f\left[x_g(n),y(n)/a_0(n),x(n)\right]\mathrm{d}a_0\,\mathrm{d}x$$

It can be shown that

$$\begin{aligned}\Phi\left[z(n)/a_0(n),x(n)\right]=&\int\limits_{-\infty}^{0}\int\limits_{\frac{z(n)}{y(n)}}^{\infty} f_4^*\left\{x_g(n)+F\left[x(n)-a_0(n)\right]/x(n)\right\}\\&\times\delta\left\{y(n)-\varphi\left[x(n)-a_0(n)\right]\right\}\mathrm{d}y(n)\mathrm{d}x_g(n)\\&+\int\limits_{0}^{\infty}\int\limits_{-\infty}^{\frac{z(n)}{y(n)}} f_4^*\left\{x_g(n)+F\left[x(n)-a_0(n)\right]/x(n)\right\}\delta\left\{y(n)-\varphi\left[x(n)-a_0(n)\right]\right\}\mathrm{d}y(n)\mathrm{d}x_g(n)\end{aligned}$$

Then the distribution density of $L(n)$ with respect to $X(n)$ and $A_0(n)$ is

$$\begin{aligned}f\left[z(n)/a_0(n),x(n)\right]&=\frac{\partial}{\partial z}\Phi\left[z(n)/a_0(n),x(n)\right]\\&=\int\limits_{-\infty}^{0}\frac{1}{y(n)}f_4^*\left\{\frac{z(n)}{y(n)}+F\left[g(n)/x(n)\right]\right\}\delta\left[y(n)-\varphi(g(n))\right]\mathrm{d}y(n)\\&=\int\limits_{0}^{\infty}\frac{1}{y(n)}f_4^*\left\{\frac{z(n)}{y(n)}+F\left[x(n)-a_0(n)\right]/x(n)\right\}\delta\left\{y(n)-\varphi\left[x(n)-a_0(n)\right]\right\}\mathrm{d}y(n)\\&=\int\limits_{-\infty}^{\infty}\frac{1}{|y(n)|}f_4^*\left\{\frac{z(n)}{y(n)}+F\left[x(n)-a_0(n)\right]/x(n)\right\}\delta\left\{y(n)-\varphi\left[x(n)-a_0(n)\right]\right\}\mathrm{d}y(n)\\&=\frac{1}{\left|\varphi\left[x(n)-a_0(n)\right]\right|}f_4^*\left\{\frac{z(n)}{\varphi\left[x(n)-a_0(n)\right]}+F\left[x(n)-a_0(n)\right]/x(n)\right\}\end{aligned}$$

The cumulative distribution of the random value $A_0(n+1)$ with respect to $X(\bullet)$ has the following form:

$$\Phi[a_0(n+1)/x(n)] = \iint \Phi[z(n)/a_0(n), x(n)] f[a_0(n)/x(n)] \mathrm{d}a_0 \, \mathrm{d}z(n)$$

Since

$$f[a_0(n)/x(n)] = f[a_0(n)]$$

then

$$\Phi[a_0(n+1)/x(n)] = \int_{-\infty}^{\infty} \int_{-\frac{a_0(n+1)-a_0(n)}{2K^*}}^{\infty} f_n[a_0(n)] \frac{1}{|\varphi[x(n)-a_0(n)]|} f_4 \times \left\{ \frac{z(n)}{y(n)} + F[x(n)-a_0(n)]/x(n) \right\} \mathrm{d}a_0(n) \mathrm{d}z(n)$$

Consequently,

$$f_{n+1}[a_0(n+1)/x(n)] = \frac{\partial}{\partial a_0(n+1)} \Phi[a_0(n+1)/x(n)]$$

$$= \frac{1}{2K^*} \int_{-\infty}^{\infty} f_n[a_0(n)] \frac{1}{|\varphi[x(n)-a_0(n)]|} f_4 \left\{ \frac{a_0(n+1)-a_0(n)}{-2K^* \varphi[x(n) a_0(n)]} + F[x(n)-a_0(n)]/x(n) \right\} \mathrm{d}a_0(n)$$

One obtains finally

$$f_{n+1}[a_0(n+1)] = \frac{1}{2K^*} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{|\varphi[x(n)-a_0(n)]|} f_n[a_0(n)] \times f_4 \left\{ \frac{a_0(n+1)-a_0(n)}{-2K^* \varphi[x(n)-a_0(n)]} + F[x(n)-a_0(n)]/x_n \right\} \mathrm{d}a_0(n) \mathrm{d}x(n) \tag{12.18}$$

In the limiting case at $n \to \infty$,

$$f_1(a_0) = \frac{1}{2K^*} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{f_1(\xi)}{|\varphi(x-\xi)|} f \left[\frac{a_0 - \xi}{-2K^* \varphi(x-\xi)} + F(x-\xi), x \right] \mathrm{d}x \, \mathrm{d}\xi$$

This is a homogeneous Fredholm integral equation of the second kind. It can be solved numerically in the general case. A non-negative function is integrated in the expression for $f_{n+1}[a_0(n+1)]$. Then $f_{n+1}[a_0(n+1)] \geq 0$. It is evident that at $n = 0$

$$\int_{-\infty}^{\infty} f_0[a_0(0)] \mathrm{d}a_0(0) = \int_{-\infty}^{\infty} \delta[a_0(0) - \hat{a}_0] \mathrm{d}a_0[0] = 1$$

where $\widehat{a}_0$ is some given threshold value.

Let us assume that for $f_n[a_0(n)]$

$$\int_{-\infty}^{\infty} f_n[a_0(0)]\mathrm{d}a_0(n)=1$$

Let us show that in this case

$$\int_{-\infty}^{\infty} f_{n+1}[a_0(n+1)]\mathrm{d}a_0(n+1)=1$$

$$Y^0=\int_R f_{n+1}[a_0(n+1)]\mathrm{d}a_0(n+1)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{1}{2K^*\left|\varphi[x(n)-a_0(n)]\right|}f_n[a_0(n)]$$
$$\times f\left\{\frac{a_0(n+1)-a_0(n)}{-2K^*\varphi[x(n)-a_0(n)]}+F[x(n)-a_0(n)]/x(n)\right\}\mathrm{d}x(n)\mathrm{d}a_0(n)\mathrm{d}a_0(n+1)$$

Let us make the change of variables

$$\varepsilon(n)=\frac{a_0(n+1)-a_0(n)}{-2K^*\varphi[x(n)-a_0(n)]}+F[x(n)-a_0(n)]$$

$$\mathrm{d}\varepsilon(n)=\frac{\mathrm{d}a_0(n+1)}{2K^*\varphi[x(n)-a_0(n)]}$$

Then

$$Y^0=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f_n[a_0(n)]f[\varepsilon(n)/x(n)]\mathrm{d}\varepsilon(n)\mathrm{d}a_0(n)\mathrm{d}x(n)$$
$$=\int_{-\infty}^{\infty}f_n[a_0(n)]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f[\varepsilon(n)/x(n)]\mathrm{d}\varepsilon(n)\mathrm{d}x(n)\right]\mathrm{d}a_0(n)$$

Due to the distribution density properties,

$$=\int_{-\infty}^{\infty}f_n[a_0(n)]\left[\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f[\varepsilon(n)/x(n)]\mathrm{d}\varepsilon(n)\mathrm{d}x(n)\right]\mathrm{d}a_0(n)$$

and due to the assumption

$$a\int_{-\infty}^{\infty}f_n[a_0(n)]\mathrm{d}a_0(n)=1$$

Consequently,

$$Y^0 = \int_{-\infty}^{\infty} f_{n+1}\left[a_0(n+1)\right] \mathrm{d}a_0(n+1) = 1$$

that was required to be proven.

The similar expressions can be obtained for the cases $m_n = \text{const}$ and $N \neq 1$, as well as for the cases of more complex multilayer neural network structures. But the complexity of the obtained expressions sharply increases. The analysis of such expressions in the explicit form has no sense. It is necessary to perform the transformation to the distribution of the correct recognition probability by means of the adjustment coefficient space integration. This is a rather complex task. One can only write general expressions for the mathematical expectation and variance of the average risk function:

$$MR = \int_A f^{**}(\mathbf{a}) \left[\int_E \int_X f_\varepsilon(\varepsilon) f(\mathbf{x}/\varepsilon) l\left[x_k = P(\mathbf{x}), \varepsilon\right] \mathrm{d}\varepsilon \, \mathrm{d}x \right] \mathrm{d}\mathbf{a}$$

$$DR = \int_A f^{**}(\mathbf{a}) \left[R - MR\right]^2 \mathrm{d}\mathbf{a}$$

The aforementioned analytical investigation complexity of the closed-loop systems with the fixed structure results in the requirement to use statistical modeling methods.

Literature

[12-1] Galushkin AI (1969) Methods of synthesis of the open-loop learning systems for nonstationary pattern recognition. Abstr. Of the III Ukr. Rep. Conf. on bionics. Kiev

[12-2] Galushkin AI, Vasilkova TA, Slobodeniuk VA, Tyukhov BP (1971) Analysis of the nonstationary pattern recognition system dynamics. Proc. MIEM 23

[12-3] Vanyushin VA, Galushkin AI, Tyukhov BP (1972) Design and investigation of multilayer pattern recognition systems. In: Some problems of biological cybernetics. Acad. Berg AI (ed) Moscow, Nauka, pp 315–323

[12-4] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energiya

[12-5] Galushkin AI, Koudryavtsev AM (1976) Matrix inversion using a multilayer system of threshold elements. Cybernetics and computer systems 33, Kiev, Naukva doumka

[12-6] Galushkin AI, Zotov Yu Ya, Shikunov Yu Ya (1972) In-line processing of experimental information. Moscow, Energiya, 360 p

[12-7] Sterity R, Coleman J, Fiddy MA (1990) Aneral network based matrix inversion algorithm. San Diego, Calif., IJCNN 1:467–470

[12-8] Sterity R, Coleman J, Fiddy MA (1991) Regularized matrix inversion on a neural network architecture. Seattle, Wash., IJCNN – 91 2:938

[12-9] Viktorov NV, Galushkin AI (1976) Design and investigation of pattern recognition systems under the arbitrary teacher qualification. Medical radio-electronics. VNII of International technology, pp 95–106

Chapter 13

Synthesis of Multilayer Neural Networks with Flexible Structure

The design of the multilayer neural networks with fixed structure and closed-cycle adjustment does not require some a priori information about the input signal, as opposed to the case of the open-cycle adjustment. However, the probability of correct recognition is restricted in the former case by the neural network structure fixation. This chapter deals with the neural network synthesis with flexible structure (Fig. 13.1) that is selected in the adjustment process.

Function $y(x)$ in Fig. 13.1 is the structure of the neural network open-loop part. Methods of adjustment for the multilayer neural network with flexible structure selected on the basis of a given probability of correct recognition include the successive neuron layer learning.

13.1 Sequential Learning Algorithm for the First Neuron Layer of the Multilayer Neural Network

Sequential learning algorithms for the first layer of the multilayer neural network are based on the gradual increase of the hyperplane number. These hyperplanes form the resultant hypersurface up to the achievement of the required recognition quality or some other condition for the learning process termination. The learning process is reduced to the formation of the logical tree. Geometric interpretation is the following. The feature space is optimally divided into two parts by some fixed structure neural network. Then the obtained subspaces are divided again, and so on.

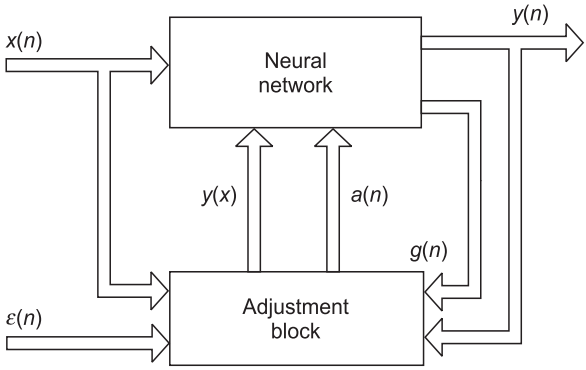


Fig. 13.1. Block diagram of the neural network with flexible structure and closed-cycle adjustment

Figures 13.2–13.4 show respectively the general block diagram of the sequential algorithm for the design of the piecewise linear divisional surface realized by the flexible structure neural network and a logical tree describing the design of the piecewise linear divisional surface. In Fig. 13.2: *I* – unit of the fixed structure neural network parameter calculation; *II* – unit of the input learning sample partition; and *III* – adjustment algorithm of the flexible structure neural network at the first step. The resultant border between two classes is shown in Fig. 13.4 by the double line. The first hyperplane $\varphi_0(x)$ divides the feature space Φ_0 into two sub-regions Φ_1 (first class patterns) and Φ_2 (second class patterns). The learning sample L_0 is divided into two ones: L_1 (vectors from Φ_1) and L_2 (vectors from Φ_2). The numbers of incorrectly classified patterns in each of the sub-regions are θ_1 and θ_2. The maximal element from the set $\{\theta_1, \theta_2\}$ is selected, and the corresponding sub-region is further divided. Let us assume

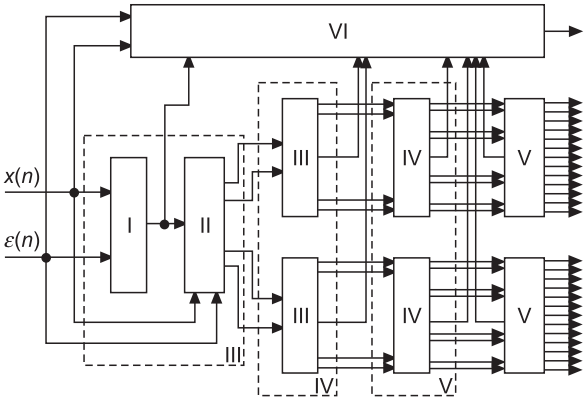


Fig. 13.2.
Block diagram of the sequential algorithm for the design of the piecewise linear divisional surface

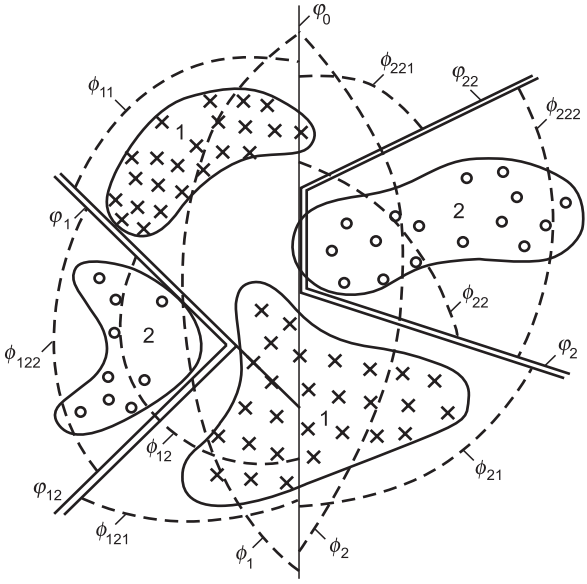


Fig. 13.3.
Drawing of the piecewise linear divisional surface

that $\theta_1 > \theta_2$. Then regions Φ_{11} and Φ_{12} are obtained after partitioning of Φ_1 by the hyperplane. Then θ_{11} and θ_{12} are calculated and the recognition errors are compared. If $\theta_1 > \theta_{11} + \theta_{12}$, then the introduction of the new hyperplane improves the recognition quality. In this case, the sample L_1 is divided into sub-samples L_{11} and L_{12}. Then the process is repeated. As a result, one obtains the set of regions $\Phi_i, \Phi_{i,j}, \ldots, \Phi_{i,j,k,\ldots,t}$, where indexes $i, j, k, \ldots, t$ take the values 1 and 2. If the drawing of the hyperplane in the subregion $\Phi_{i,j,k,\ldots,t}$ does not improve the recognition quality, then the partitioning of the obtained regions must be continued. The number of steps decreasing the recognition

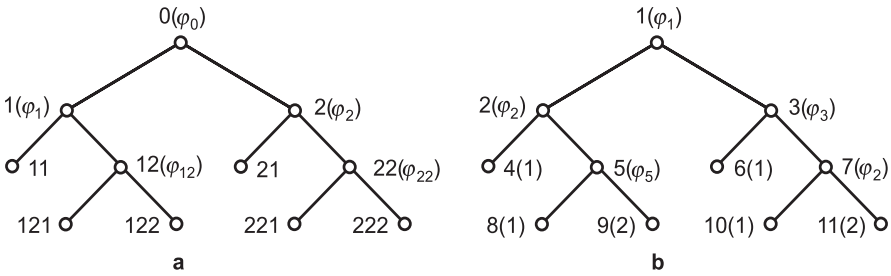


Fig. 13.4. Logical tree: **a** scheme of the drawing of the piecewise linear divisional surface shown in Fig. 13.2; **b** sequential numeration of the tree knots

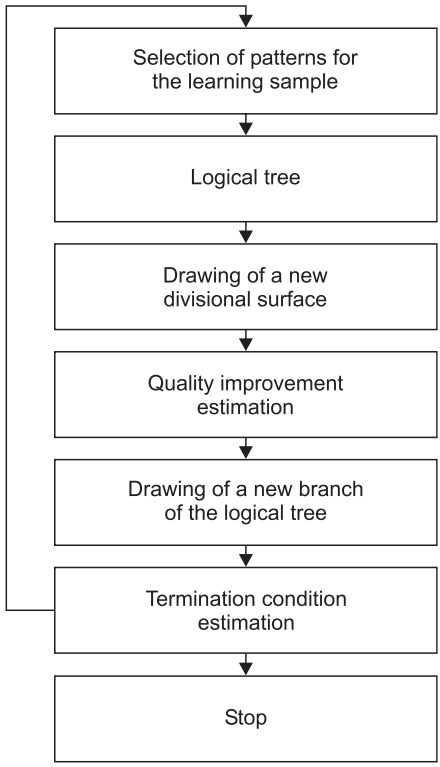

Selection of patterns for the learning sample
Logical tree
Drawing of a new divisional surface
Quality improvement estimation
Drawing of a new branch of the logical tree
Termination condition estimation
Stop

Fig. 13.5. The block diagram of the program realizing the algorithm of the piecewise linear divisional surface drawing

quality must be limited in the design of the similar algorithms [13-1]. If the error does not decrease at the given number of steps, then the initial region $\Phi_{i,j,k,\ldots,t}$ is excluded. The following rules for the algorithm termination are considered in [13-1]: (1) Termination at the given value of the error probability; (2) Termination at the given value of the hyperplane number (the number of neurons in the first layer). The block diagram of the program realizing the algorithm of the piecewise linear divisional surface drawing is shown in Fig. 13.5. Let us describe the operators that are not clear from the previous description.

Operator "Logical tree". As it is seen in Fig. 13.4, the tops of the tree are of two types: intermediate tops and terminations. The tree root is the knot with index 0, and terminations correspond to certain pattern classes. Each pattern x after the pass of the operator "Logical tree" enters one of the terminations. Function $\varphi_{i,j,k,\ldots,t}(x)$ is used for making a decision about further movement direction from the top $i, j, k, \ldots, t$. If $\varphi_{i,j,k,\ldots,t}(x) \geq 0$ then the movement is to the right branch and vice versa. The logical tree in Fig. 13.4a takes the form shown in Fig. 13.4b under the sequential numeration of the tree tops. The logical tree is convenient for describing the following with the three-column matrix:

$$\mathbf{C} = \begin{bmatrix} 0 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 6 & 7 \\ 1 & 0 & 0 \\ 0 & 8 & 9 \\ 1 & 0 & 0 \\ 0 & 10 & 11 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 0 \end{bmatrix}$$

The s-th row of matrix $\mathbf{C}$ corresponds to the s-th top of the logical tree. The matrix rows, similar to the tops, are of two types. The row $(0\ s\ s+1)$ describes the intermediate top of the tree. One takes the divisional surface $\varphi_s = 0$ corresponding to this top, and the transfer to the top s is performed if sign $\varphi_s = -1$ or to the top $s+1$ if sign $\varphi_s = 1$. If the row has the form $(k\ 0\ 0)$, where $k = 1,2$, then it describes one of the tree terminations. If the point x_j appears in such a point after the sequential use of several divisional surfaces, then it belongs to the class A_k. The drawing of the new hyperplane $\varphi_i(x)$ results in the appearance of two new tree branches going from the top i. Matrix with U rows obtains two new rows $(U+1)$ and $(U+2)$ of the following form:

$U+1$: 1 0 0

$U+2$: 2 0 0

and the i-th row gets the record

0 $U+1$ $U+2$

i.e., the i-th top now becomes the intermediate top of the logical tree.

Operator "Drawing of a new divisional surface" can use any neuron adjustment algorithm described in Chap. 9. It can realize any fixed structure neural network described in Chap. 9 and 10.

Operator "Quality improvement estimation" is used for the recognition quality improvement estimation. The results of its application are used for the logical tree design: the improvement of the quality leads, for example, to the division of the region with the highest value of the average risk function. In the opposite case, the division of the lastly obtained regions takes place.

13.2 Learning Algorithm for the First Neuron Layer of the Multilayer Neural Network Using the Method of Random Search of Local and Global Function Extrema

This algorithm was designed on the basis of methods described in Chap. 8. One can ignore in this case the tree structure design, and all the neurons providing local extrema of the average risk function are included in the first layer (Figs. 13.6, 13.7).

Four hyperplanes in the two-dimensional feature space shown in Fig. 13.6 determine four local extrema of the average risk function. Figures inside the circles correspond to the numbers of the logical function arguments in each region of the multidimensional feature space. Table 13.1 gives the logical function values in the example shown in Fig. 13.6. Tick marks correspond to the values that are not defined at the given argument values. Zero index corresponds to the regions without patterns.

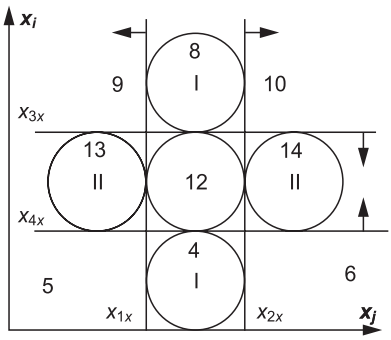


Fig. 13.6.
Illustration of the learning method for the first neuron layer of the multilayer neural network using an algorithm of random search

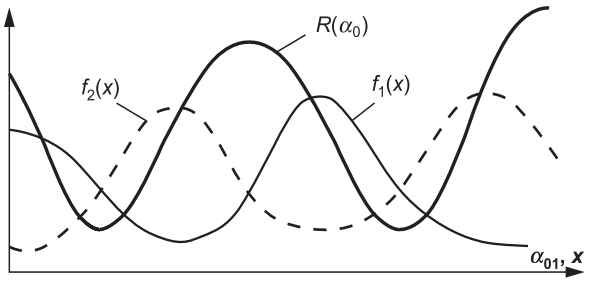


Fig. 13.7.
Multi-extremum property of the average risk function under multi-modal distributions $f(x/\varepsilon)$

Table 13.1. Logical function values in the example of in Fig. 13.6

z	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
y_1	−1	1	−1	1	−1	1	−1	1	−1	1	−1	1	−1	1	−1	1
y_2	−1	−1	1	1	−1	−1	1	1	−1	−1	1	1	−1	−1	−1	−1
y_3	−1	−1	−1	−1	1	1	1	1	−1	−1	−1	−1	1	1	1	1
y_4	−1	−1	−1	−1	−1	−1	−1	−1	1	1	1	1	1	1	1	1
ε	*	*	*	*	−1	0	0	*	−1	0	0	*	0	1	1	*

It is evident that deterministic search methods do not provide the local extremum overrunning. The solution is the introduction of the random elements into the search procedure.

The main algorithm stages are the following:

a The adjustment coefficient vector components of the current neuron are randomly selected;
b The next local extremum of the average risk function is found by some neuron learning method;
c This extremum is recorded if it was not found earlier.

The transfer to the first stage is performed after the termination of the third one, and the adjustment coefficient vector of the next first layer neuron is determined.

The experimental investigation of one performance cycle of such a learning algorithm for the first layer neuron was described in Chap. 12. Figure 13.8 shows the block diagram of the program that realizes this algorithm.

The plan of experiments with this program was aimed at the analysis of the learning process properties. The input signal characteristics and the adjustment algorithm were similar to those described in Sect. 12.2. The following characteristics must be analyzed:

1. The experimental estimation of the random procedure convergence;
2. Dependence of the total calculating time on the feature space dimensionality N, number of extrema U, and step value Δ. The learning procedure is performed until the sequential random initial condition injection results in the search of all the quality functional local extrema under the given modality of the distribution functions of the input signal. The number of the random search steps required for the search of all the local extrema is given in Table 13.2, where U is the number of the desired extrema, and η_U is the number of the random procedure steps required for the search of all the U extrema. The approximate estimations for the mathematical expectation and variance of the step number have the form (8.16a).

Table 13.2 and the data described in Sect. 8.6 provide the estimation of the total learning time under the particular modality of the input signal. This time increases linearly with the feature space dimensionality increase.

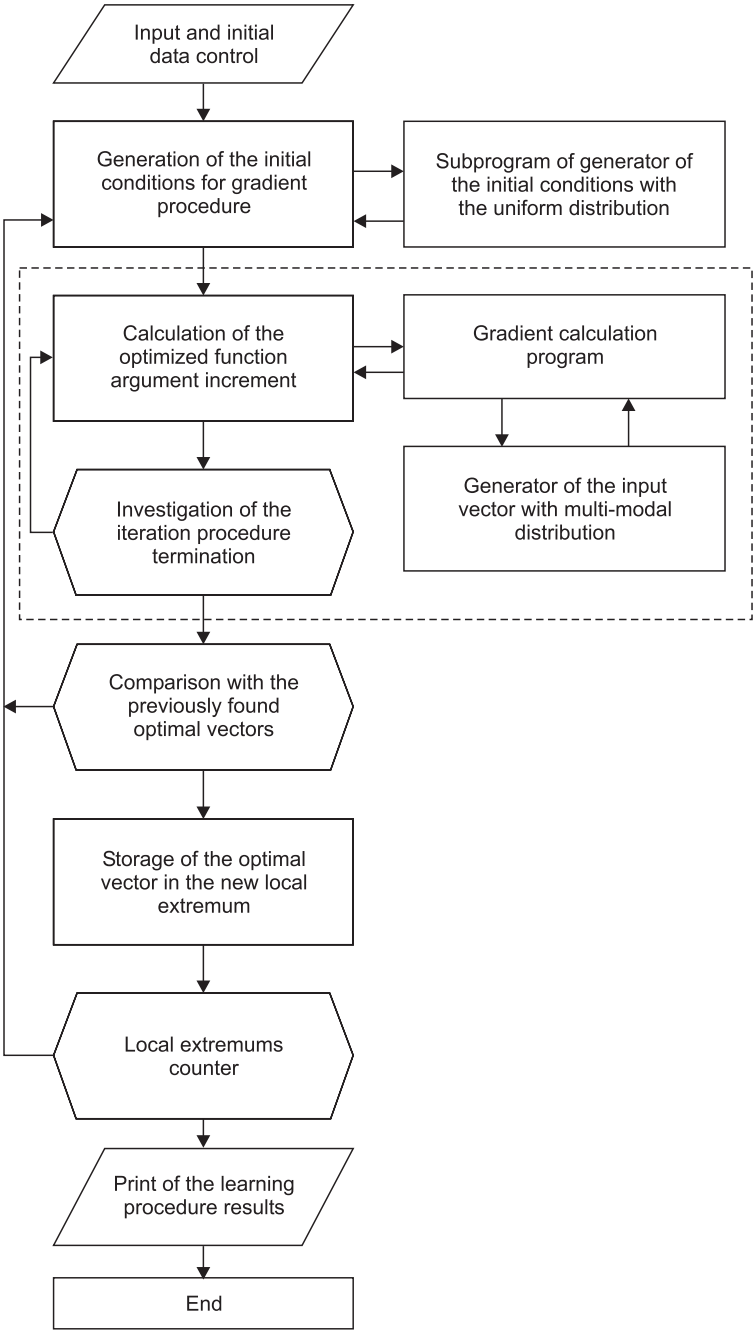

Input and initial data control
Generation of the initial conditions for gradient procedure
Subprogram of generator of the initial conditions with the uniform distribution
Calculation of the optimized function argument increment
Gradient calculation program
Investigation of the iteration procedure termination
Generator of the input vector with multi-modal distribution
Comparison with the previously found optimal vectors
Storage of the optimal vector in the new local extremum
Local extremums counter
Print of the learning procedure results
End

Fig. 13.8. Block diagram of the program realizing learning algorithm for the first neuron layer of the multilayer neural network using random search methods

Table 13.2.
Number of random search steps required for the search of all local extrema

U	η_U	$M\eta_U$	$D\eta_U$
1	1	1	–
2	4	3	2
3	8	6	3
5	8	10	6
7	23	14	8
10	33	22	12

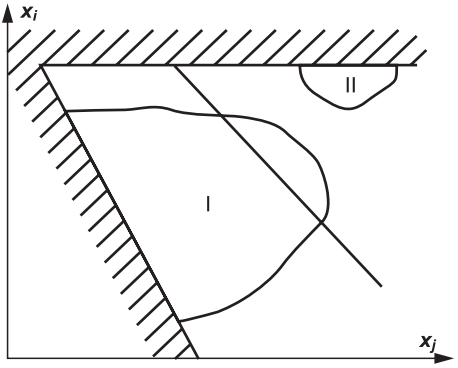


Fig. 13.9.
Illustration of the process for the increase of the error probability at some step of the sequential algorithm performance: *I* – the first class; *II* – the second class

13.3 Analysis of Algorithm Convergence under the Hyperplane Number Increase

The algorithm convergence under the neural network structure complication depends on the rules for the selection of the sub-region for partitioning and the learning algorithm at each partition step. The method described above of the selection of the sub-region for partitioning is optimal by the algorithm convergence rate. The simplified methods of the feature space sequential partitioning usually used in practice and consisting in the neuron open-cycle adjustment using the initial learning samples moments often result in the increase of the error probability at some step of the algorithm performance (Fig. 13.9).

The non-shaded area in Fig. 13.9 corresponds to the next partition region, and the divisional surface is drawn perpendicular to the line linking the centers of two classes. The error at this partition step increased because some patterns of the first class appeared to be classified as second class patterns. To provide the stable decrease of the error probability, one must use the closed-cycle adjustment with the second discrete error distribution moments α_{2g} minimization at each procedure step. This provides the minimum number of the first layer neurons. However, it is sometimes necessary to increase the number of the first layer neurons voluntarily at the expense of the sharp simplification of the neuron learning algorithm.

The increase of the hyperplane number in the most unfavorable case results in the error probability estimation convergence to zero due to the finite learning sample length.

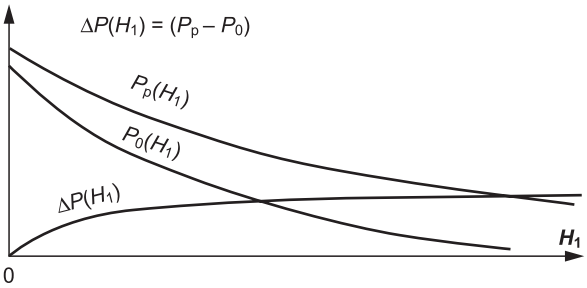


Fig. 13.10.
The analysis of algorithm convergence under the hyperplane number increase on the learning and recognition stages

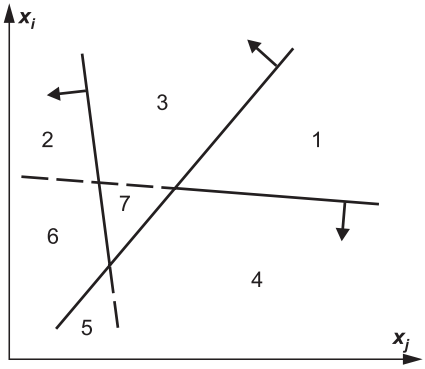


Fig. 13.11.
Formation of the learning sample at the first layer neuron output

Two stages of the neural network design therefore exist: the algorithm learning stage and the algorithm precision estimation stage. At the sample length equal to M, only its minority part M_1 is used for the algorithm learning. The trained algorithm is then used for the recognition performance on the sample part $M_2 = M - M_1$, and the real algorithm precision is estimated by the recognition probability error $P_p(H_1)$. Function $\Delta P_p(H_1) = P_p(H_1) - P_0(H_1)$ shown in Fig. 13.10 must in principle increase with the increase of the hyperplane number due to the decrease of the algorithm capability for generalization. Here $P_0(H_1)$ is the function of the error probability change on the neural network learning stage. Function $P_p(H_1)$ often has a local minimum under the given finite value of H_1, amounting, for example, to H_1'. It is recommended to use namely this number H_1' of hyperplanes if $P_p(H_1')$ satisfies the initial conditions.

The particular result of the first layer neuron learning in the case of the multilayer neural network with two solutions is the logical function that determines the sequence of multidimensional feature space partitioning. This logical function is sometimes not defined not only on the complete sets of arguments but on some separate arguments. The simplest illustration of underdefiniteness of such a kind of logical functions is represented in Fig. 13.11 and Table 13.3. The Roman numerals indicate the initial regions of the feature space for the formation of some set of the logical function $\varepsilon(\mathbf{y})$ arguments. The cells marked by the tick marks indicate variable sets that never appear at the output of the first neuron layer. The cells marked by the sign $\oplus$ indicate the variable values out of the complete set of 2^{H_1} values that also never appear at the output of the first neuron layer. The procedure of the sequential partitioning shown in Fig. 13.11

can be illustrated by the tree and matrix with the form represented in Fig. 13.12. Here *I–IV* are the regions obtained as a result of the sequential partitioning.

The problem of the extension of a definition of the logical function $\varepsilon(\mathbf{y})$ emerges in connection with the necessity to form the arrays of the learning vectors at the output of the first layer neurons required for the adjustment of the following neuron layers. The main problem here is to extend the logical function definition onto the partially given sets of its arguments. The definition extension onto set 8 (Fig. 13.11, Table 13.3) is not necessary because this set never appears in this particular task of the piecewise linear divisional surface design. The definition extension is carried out in the following way. Vectors with existent coordinates, initial teacher instruction, and complete sweeping across the absent variable values are recorded in the learning array for the second layer neurons of the multilayer neural network represented in Table 13.4.

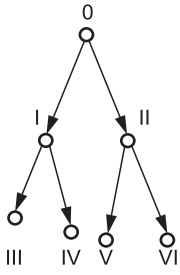
The logical function for the adjustment of all the neuron layers except the first one is formed in Table 13.4.

Table 13.3. Underdefiniteness of a logical function

Region number		1	2	3	4	5	6	7	8
ε		–1	1	–1	1	1	1	–1	*
y	First neuron	–1	1	1	–1	–1	1	1	*
	Second neuron	⊕	1	–1	⊕	⊕	1	–1	*
	Third neuron	–1	⊕	⊕	1	1	⊕	⊕	*

Table 13.4. Learning array for the second layer neurons

Region number		1'	1"	2'	2"	3'	3"	4'	4"	5'	5"	6'	6"	7'	7"
ε		–1	–1	1	1	–1	–1	1	1	–1	–1	1	1	–1	–1
y	First neuron	–1	–1	1	1	1	1	–1	–1	–1	1	1	1	1	1
	Second neuron	–1	1	1	1	–1	–1	–1	1	1	1	1	1	–1	–1
	Third neuron	–1	–1	–1	1	–	1	1	1	1	–1	–1	1	–1	1



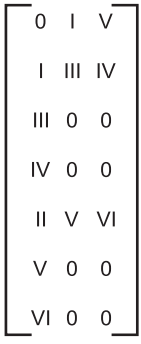


Fig. 13.12. Logical tree and matrix of transformations for the example represented in Fig. 13.11

13.4
Learning Algorithms for the Second Layer Neurons of the Two-Layer Neural Network

13.4.1
Condition of the Logical Function $\boldsymbol{\varepsilon}(\mathbf{y})$ Realizability Using One Neuron

The goal of this section is to test the logical function realizability using one second layer neuron of the two-layer pattern recognition system. If the result of this test is negative, then the transfer to the synthesis of a three-layer neural network must be performed.

$$(1)\ \sum_{z=1}^{Z}\left|\mathbf{c}^T\mathbf{y}(z)\right|\ ;\quad (2)\ (\mathbf{c}^T\mathbf{b})$$

Figure 13.13 is the illustration of the logical function realizability using one neuron. Here, the value of the neuron output analogous signal $g(n)$ is less than zero across all sets of the input binary variables $y(z)$ (z is the number of the set) of the first class and less than zero across all sets of the input binary variables of the second class. The value $\Delta g = g^{(2)}_{\min} - g^{(1)}_{\max}$ is termed "interval" [13-5]. The necessary and sufficient condition for the logical function realizability using one neuron can be written in the following form:

$$\left.\begin{aligned}&\varepsilon(z) - \operatorname{sign} g(z)\\ &g(z)\varepsilon(z) = |g(z)|\end{aligned}\right\} \tag{13.1}$$

The summation of the first members and second members of equations (13.1) gives the condition for the logical function realizability using one neuron in the following form:

$$\sum_{z=1}^{Z} g(z)\varepsilon(z) = \sum_{z=1}^{Z}|g(z)| \tag{13.2}$$

or in another form

$$\sum_{j=1}^{H_1} a_j \sum_{z=1}^{Z} y_j(z)\varepsilon(z) + a_0 \sum_{z=1}^{Z}\varepsilon(z) = \sum_{z=1}^{Z}\left|\sum_{j=1}^{H_1} a_j y_j(z) + a_0\right| \tag{13.3}$$

The expressions

$$\sum_{z=1}^{Z} y_j(z)\varepsilon(z)\ ,\quad \sum_{z=1}^{Z}\varepsilon(z)$$

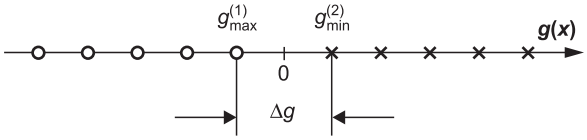


Fig. 13.13. The test of the logical functions realizability using one neuron

are unambiguously determined by the given logical function and can be calculated before the solution of the problem for the synthesis of the neuron realizing this logical function. Similarly to [13-5], let us introduce the expression

$$b_j = \sum_{z=1}^{Z} y_j(z)\varepsilon(z) \quad , \quad j = 1,\ldots,H_1 \tag{13.4}$$

Notice that

$$b_0 = \sum_{z=1}^{Z} \varepsilon(z) \quad , \quad \text{because} \quad x_{0k}(z) = 1 \quad ; \quad z = 1,\ldots,Z$$

Expressions (13.3) and (13.4) give

$$\mathbf{a}^T\mathbf{b} = \sum_{j=1}^{H_1} a_j b_j = \sum_{z=1}^{Z} \left| \sum_{j=1}^{H_1} a_j y_j(z) + a_0 \right|$$

or

$$\mathbf{a}^T\mathbf{b} = \sum_{z=1}^{Z} |g(z)| \tag{13.5}$$

The expression (13.5) gives the necessary and sufficient condition for the logical function $\varepsilon(y)$ realizability.

Let the logical function $\varepsilon(z)$ determined in Z points of H_1-dimensional binary argument $\mathbf{y}(z)$ not be realizable using only one neuron with the weighting coefficient vector **a**. The scalar product of vector **a** and characteristic vector of the logical function is less than the sum of absolute values of the neuron output analogous signal across all $z = 1,\ldots, Z$. Consequently, the weighting coefficient vector of the neuron realizing the given logical function with characteristic vector **b** must minimize (to zeroth value) the following functional:

$$I(\mathbf{a}) = \sum_{z=1}^{Z} |g(z)| - \mathbf{a}^T\mathbf{b} \tag{13.6}$$

Vector **b** is to some sense close to the weighting coefficient vector **a** of the neuron realizing the logical function that corresponds to **b**. The difference

$$\varepsilon(z) - \frac{1}{2^{N_1}}\mathbf{c}^T\mathbf{x}_k(z)$$

can be regarded as an error of the logical function realization [13-5]. Then the mean-root-square error is minimal at $\mathbf{c} = \mathbf{b}$. Consequently, the corresponding vector **b** can sometimes be taken as a weighting coefficient vector **a** realizing the logical function. However, in general, vector **b** cannot be always taken as vector **a**.

The expression (13.5) is similar to the expression (13.2). The latter one can be represented as a system of linear inequalities and the former one as a nonlinear equation. The use of (13.5) is slightly simplified because the initial logical function $\varepsilon(\mathbf{y})$ defined in 2^{H1} points of H_1-dimensional space of binary (–1, 1) variables is represented in (13.5) by an H_1-dimensional analogous vector, whereas in (13.2) it is represented by 2^{H1} binary numbers.

13.4.2
Synthesis of a Neuron by the Functional Minimization Method

The aforementioned correspondence between (13.2) and (13.5) indicates the advantages of (13.5). However, the complexity of the explicit expression for the nonlinear term

$$\sum_{z=1}^{Z}\left|g(z)\right|$$

emerges here. This complexity can be overcome by the use of the appropriate approximation. According to (13.6), the minimized functional has the following form:

$$I(\mathbf{c},\mathbf{b})=\sum_{z=1}^{Z}\left|\mathbf{c}^T\mathbf{y}(z)\right|-\mathbf{c}^T\mathbf{b} \tag{13.7}$$

Here **c** is the arbitrary weighting coefficient vector that provides a non-zero analogous error of the neuron; **b** is the characteristic vector of the given logical function. The following condition is assumed at the determination of the vector providing a minimum of (13.7): either vector **a** = **c** is the weighting coefficient vector realizing the given logical function, or the given logical function cannot be realized with the use of only one neuron.

Figure 13.14 shows conditionally the dependencies of summands in (13.7) upon c_i for realizable and non-realizable logical functions with the use of only one neuron: *1* – physically realizable logical function; *2* – non-realizable logical function.

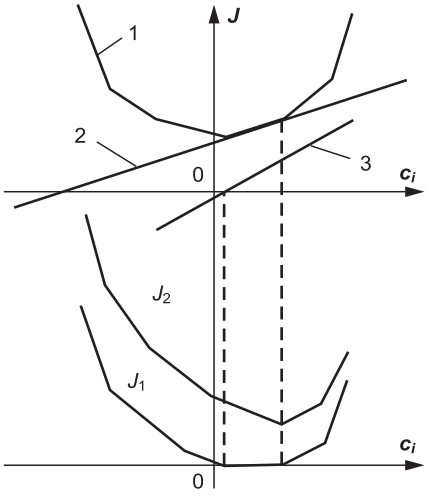


Fig. 13.14.
General form of the quality functional for physically realizable and non-realizable logical function with the use of only one neuron

The described synthesis method for the second layer of the multilayer neural network is based on the following representation:

$$q|g(z)| \approx \xi_2[q^2 g^2(z)] + \xi_4[q^4 g^4(z)] + \ldots$$

where q is the normalizing factor limiting the approximation region in the following way: $1 \le q|g(z)| < 0$. The approximation of $q|g(z)|$ by k terms is the k-th order approximation.

In the case of $k = 1$,

$$|g(z)| \approx \xi_2[qg^2(z)]$$

Consequently,

$$\sum_{z=1}^{Z} |g(z)| \approx \xi_2 q \sum_{z=1}^{Z} g^2(z)$$

or

$$\sum_{z=1}^{Z} |g(z)| = \xi_2 q \sum_{z=1}^{Z} \sum_{i=0}^{H_1} \sum_{j=0}^{H_1} c_i c_j y_i(z) y_j(z)$$

$$\sum_{z=1}^{Z} |g(z)| = \xi_2 q \sum_{i=0}^{H_1} \sum_{j=0}^{H_1} c_i c_j \sum_{z=1}^{Z} y_i(z) y_j(z)$$

The sum

$$\sum_{z=1}^{Z} y_i(z) y_j(z) = d_{ij}$$

is totally determined by the given logical function (by its argument values) and can be calculated before the neuron synthesis solution. The same is valid for its characteristic vector. The following expression is valid for the whole set of arguments of the logical function defined in 2^{H_1} points of H_1-dimensional space of binary (–1, 1) variables:

$$\sum_{z=1}^{Z} y_i(z) y_j(z) = 2^{H_1} \delta_{ij} \tag{13.8}$$

where δ_{ij} is Kronecker symbol. This expression is not valid in the general case of the multilayer neural network synthesis. In this particular case, taking into account (13.8), one obtains

$$\sum_{z=1}^{Z} |g(z)| = \xi_2 q 2^{H_1} \sum_{l=0}^{H_1} c_l^2 \tag{13.9}$$

In the general case,

$$\sum_{z=1}^{Z}|g(z)| = \xi_2 q\left(\mathbf{c}^T D\mathbf{c}\right) \;\; ; \quad D = \left[d_{ij}\right] \tag{13.10}$$

If (13.9) takes place, then

$$I(\mathbf{c},\mathbf{b}) = \xi_2 q 2^{H_1} \sum_{l=0}^{H_1} c_l^2 - \sum_{l=0}^{H_1} c_j b_j$$

The expression for the optimal vector **c** providing minimum $I(0)$ has the following form:

$$c_i \approx P_i b_i$$

where

$$P_i = \left(\xi_2 q 2^{H_1+1}\right)^{-1}$$

The realizability of the logical function using one neuron is invariant with respect to the multiplication of a_i by the constant coefficient. Then the expression for the desired weighting coefficient vector realizing the logical function with the characteristic vector **b** under condition (13.8) has the following form:

$$a_i \approx b_i \;\; , \quad i = 0, \ldots H_1 \tag{13.11}$$

It is seen therefore that at the first-order approximation under the condition (13.8), the weighting coefficient vector is equal to the logical function characteristic vector. If in this case the first-order approximation doesn't appear to be valid, then one usually assumes

$$a_i \approx b_i \;\; , \quad \text{only for} \quad i = 1, \ldots H_1,$$

and the value a_0 is varied for providing the possibility to realize the logical function using only one neuron (see an example below).

In the general case, when (13.8) is not valid

$$I(\mathbf{c},\mathbf{b}) = \xi_2 q\left(\mathbf{c}^T D\mathbf{c}\right) - \sum_{l=0}^{H_1} c_j b_j$$

and the desired weighting coefficient vector is calculated according to the equation

$$\mathbf{a} = D^{-1}\mathbf{b}$$

This is the main expression for the neuron synthesis by means of the functional minimization at the first-order approximation. Matrix D^{-1} and vector **b** are calculated by the initial values of the realized logical function. The calculation of a_0 is similar to the case of the validity of (15.8).

Example. Let the divisional surface configuration be the same as that which is shown in Fig. 13.15.

Table 13.5 gives the values of the logical function of four variables. The tick mark indicates the argument values that do not participate in the formation of the initial piecewise linear divisional surface. The values of the binary input variables are ordered by the increase of the corresponding decimal numbers z. The full set of the logical function values is realized by the following transformation:

$$\varepsilon = x_1x_3 + x_3x_4 + x_1x_2x_4$$

The logical function characteristic vector is

$$b_i = \sum_{z=2}^{2^{H_1}} \varepsilon(z)x_i(z) \quad , \quad i = 0,\ldots,H_1; \;\; x_0 = 1$$

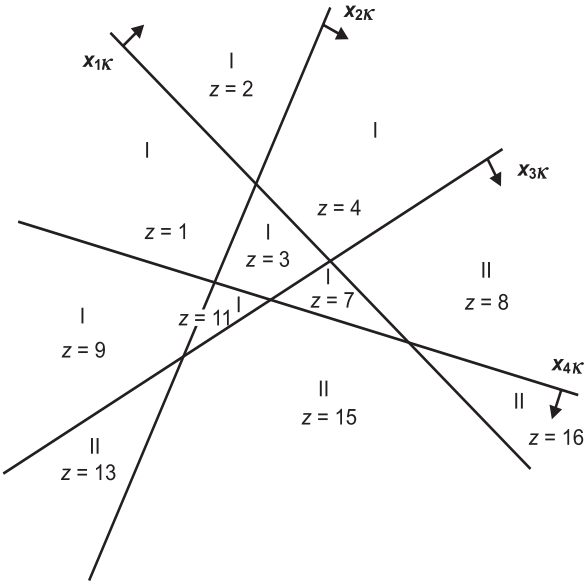


Fig. 13.15. Illustration of the synthesis of the second layer neurons in the two-layer neural network by method of the functional minimization

Table 13.5. Values of the logical function of four variables

z	0	1	2	3	4*	5*	6	7	8	9*	10	11*	12	13*	14	15
y_1	–1	1	–1	1	–1	1	–1	1	–1	1	–1	1	–1	1	–1	1
y_2	–1	–1	1	1	–1	–1	1	1	–1	–1	1	1	–1	–1	1	1
y_3	–1	–1	–1	–1	1	1	1	1	–1	–1	–1	–1	1	1	1	1
y_4	–1	–1	–1	–1	–1	–1	–1	–1	1	1	1	1	1	1	1	1
ε	–1	–1	–1	–1	–1	1	–1	1	–1	1	–1	1	1	1	1	1

In the considered example, $b_0 = -2$, $b_1 = 6$, $b_2 = -2$, $b_3 = 10$, and $b_4 = 6$. It is easy to check that these neuron coefficients provide the realization of the initial logical function using this neuron. However, in the case of the first-order approximation, the additional variation of coefficient b_0 is required. Using the calculated coefficients b_i and (13.11) for $i = 1, \ldots, H_i$, one obtains the following value (Table 13.6):

$$B(z) = \sum_{i=1}^{n} a_i y(z)$$

The sweeping of the neuron threshold ($b_0 = a_0$) values is performed per unit in the interval $[B(z)_{\max} - 0.5] \div [B(z)_{\min} + 0.5]$.

The method of the neuron synthesis by means of the functional minimization and first-order approximation at the incomplete variable set determined by the divisional surface form (Fig. 13.16, Table 13.7) can be illustrated in a similar way.

Consequently, the general procedure for the neuron synthesis by the method of functional minimization includes the following stages (step by step with dependence on the logical function realizability): (1) determination of the characteristic vector **b**; (2) determination of the threshold b_0; (3) the use of the second-order approximation, etc.

It is evident that the described method of the neuron synthesis is equivalent to the usual neural network synthesis methods with open-cycle adjustment and input signal high-order moment consideration. In the described method, at the first-order approximation, the characteristic vector is the vector of the divisional surface drawn in the middle between the centers of two classes.

Table 13.6. Results with calculated coefficients b_i and (13.11)

z	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$B(z)$	–20	–8	–24	–12	0	12	–4	8	–8	4	–12	0	12	24	8	20

Table 13.7.
Synthesis of the second layer neuron

z	0	1	2	3	4	5	6	7
y_1	1	1	1	1	1	–1	–1	1
y_2	–1	1	1	1	–1	1	–1	1
y_3	–1	–1	–1	–1	1	1	1	1
ε	1	1	1	–	–1	–1	1	–1

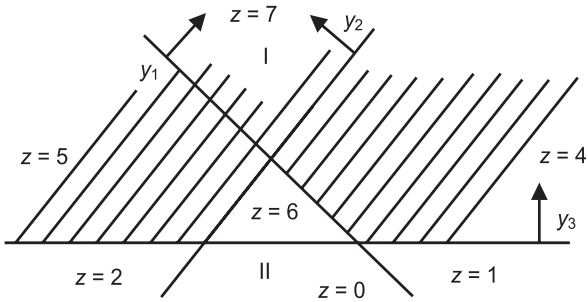


Fig. 13.16.
Illustration of the synthesis of the second layer neuron in the two-layer neural network at the nonstrictly defined logical function $\varepsilon(\mathbf{y})$

13.4.3
Neuron Synthesis by the Threshold Function Tables

The sufficiently large attention paid to the problem of the second layer neuron synthesis in the multilayer neural network with two solutions is explained by characteristic peculiarities observed in the process of operating with the first-layer neuron output signals in the binary space.

Neuron synthesis by the threshold function tables [13-5] is based on the use of the logical function characteristic vectors. This method allows one to obtain the neuron optimal parameters in the case when the one-neuron logical function realization is possible [13-5]. The method of the neuron synthesis by the threshold function tables can be used when the first-layer neuron number is not more than seven. The process of design of the characteristic vector tables and corresponding weighting coefficient vectors of the second-layer neuron is described in detail in [13-5]. The procedure consists in the following steps:

1. Vector **b** determination;
2. Formation of the decreasing sequence of $|b_i|$ values ($i = 0, \ldots, H_1$) and checking its presence in the corresponding table. If it is absent, then the given logical function is not realizable by one neuron, and the synthesis procedure is terminated;
3. If the sequence in the table is found, then the given logical function is realizable by one neuron. The weighting coefficient vector **a** can be found in the following way. Write out sequence $|a_i|$ related in the table to sequence $|b_i|$. Then make the replacements and sign changes of a_i in the precise correspondence with those made in vector **b** for its canonical representation in the table. The obtained sequence of $H_1 + 1$ elements $a_i = (i_0, \ldots, H_1)$ represents the weighting coefficients of the neural network second-layer neuron.

13.5
Learning Algorithm for Neurons of the Second and Third Layers in the Three-Layer Neural Network

The training of the second and third layers in the three-layer neural network in the case when the first layer is adjustable is equivalent to the independent learning problem for the two-layer neural network with binary input signals. This section deals with two kinds of neural network design: design in the form of a threshold-disjunctive neural network [13-5] and in the form of two neuron layers with adjustable coefficients.

The initial data in the case of the threshold-disjunctive neural network synthesis is the completely defined logical function $\varepsilon(\mathbf{y})$, and the synthesis is performed in the following order:

1. Execution of the Kwine-McKlaski procedure over the $\varepsilon(\mathbf{y})$ function until all its prime implicants are obtained;
2. Find all common intersections (centers of gravity) of two or more prime implicants and combine into wyes the prime implicants that have a common center of gravity;
3. Find characteristic vector of each wye, and check these wyes' realizability using one neuron (use any method described in the previous chapter);

4. Find all possible sub-wyes for each wye that cannot be realized using one neuron. The sub-wye is defined as the wye subset that can be realized using one neuron and that is not a subset of any other wye;
5. Add wyes and sub-wyes from p. 3, 4 to the list with prime implicants realizable by one neuron and mark the sets covered by each record from this list;
6. Select the minimum number of records covering all the units of the logical function $\varepsilon(\mathbf{y})$. The linear threshold elements that realize these records constitute either the first layer of the threshold-disjunctive network or the tandem network [13-5] equivalent to this threshold-disjunctive network.

The method of the sub-wye search includes the following procedures:

1. Determine all the implicants that have the intersection with the center of gravity of the considered wye;
2. These implicants together with the prime implicants are considered thereafter in all possible combinations, their characteristic vectors are calculated, and then the test on their one-neuron realizability is performed.

This method appears to be rather complex when the number of prime implicants is large. One can use therefore another method for the sub-wye search:

1. If the wye that cannot be realized by one neuron consists of the prime implicants, then one must consider all the groups of these prime implicants (G–1 implicants in each group) and test each group on its one-neuron realizability;
2. If at least one of such groups is one-neuron realizable then this wye is two-neuron realizable without its further partitioning;
3. If all the groups are not realizable by one neuron then the procedure must be repeated but with (G-2) prime implicants in each group;
4. The described process continues until all the prime implicants are spent. The obtained one-neuron realizable groups are the desired sub-wyes.

The synthesis procedure in the case of a nonstrictly defined logical function $\varepsilon(\mathbf{y})$ consists of the following steps:

1. Extend a definition of the logical function $\varepsilon(\mathbf{y})$ to all the variable sets where it takes the arbitrary values;
2. Perform the process of the threshold-disjunctive network synthesis described above for the case of a completely defined logical function until it is discovered that all the wyes and sub-wyes are one-neuron realizable;
3. Compose the table of prime implicants with the number of rows equal to the number of wyes, sub-wyes and prime implicants obtained at the second step of the synthesis procedure, and with the number of columns equal to the number of logical function $\varepsilon(\mathbf{y})$ sets. Thus, all the arbitrary values of $\varepsilon(\mathbf{y})$ are taken equal to (–1);
4. The minimal table record set covering all the units of $\varepsilon(\mathbf{y})$ is selected. At that point, the definition extension of its arbitrary values is automatically performed. Then the synthesis process is terminated.

The design of two output neural network layers in the form of a neural network with adjustable coefficients can be carried out on the basis of the following considerations.

The feature space in this case is binary, and the space dimensionality is equal to the number of the first-layer neurons. Therefore, training of the second-layer neurons in the three-layer neural network can be performed using any method described in Sects. 13.1 and 13.2. Then, after the second-layer neuron learning termination, the logical-tree structure of the third layer can be tested on its realizability by one third-layer neuron.

13.6 General Methods of the Multilayer Neural Network Successive Synthesis

The methods described above of successive adjustment of the three-layer neural network can be generalized for the case of multilayer neural networks in the following way:

1. The first neuron layer of the multilayer neural network is adjusted by the initial samples. The number of neurons and the values of adjustment coefficients are selected;
2. The one-neuron realizability of the obtained logical function is checked. In the positive case, the network synthesis is terminated;
3. In the negative case, the second-layer neurons are trained according to p. 1. The number of neurons and the values of adjustment coefficients are selected;
4. The one-neuron realizability of the obtained logical function is checked…, and so on similar to p. 2.

It is simple to generalize this technique for the neural network with a solution continuum. Here, the number of first-class and second-class patterns is preserved at the transfer from one layer to another. The multilayer neural network quality criterion is not only the correct recognition probability at the neural network output but also the function of this probability change during the transfer from one layer to another.

The results of the application of the described technique for the multilayer neural network synthesis are the neural network layer number, the number of neurons in each layer, and the adjustable coefficient values. It must be mentioned that the described method of the multilayer neural network learning allows one to train any structure considered in Chap. 9 instead of one neuron at each learning step.

The successive adjustment procedure can be simply generalized for the self-learning mode. The optimization criterion for the next hyperplane drawing in this case is the criterion of the specific average risk function minimum.

13.7 Learning Method for the First-Layer Neurons of a Multilayer Neural Network with a Feature Continuum

This section deals with the learning algorithm for the first-layer of a multilayer neural network with a feature continuum and the ways of its physical realization. The peculiarity of the learning process for the multilayer neural networks with a feature continuum emerges at the first-layer neuron training. The expressions for $a(\mathbf{i})$-functions and coefficients a_0 in the simplest case have the following form:

$$a(\mathbf{i}) = m_1(\mathbf{i}) - m_2(\mathbf{i})$$

$$a_0 = \frac{1}{2}\left[\int_J m_1^2(\mathbf{i})\mathrm{d}\mathbf{i} - \int_J m_2^2(\mathbf{i})\mathrm{d}\mathbf{i}\right]$$

If $x_1(\mathbf{i},n)$ and $x_2(\mathbf{i},n)$ are the sets of patterns of the first and second classes, then functions $m_1(\mathbf{i})$ and $m_2(\mathbf{i})$ are

$$m_k(\mathbf{i}) = \frac{1}{M}\sum_{n=1}^{M} x_k(\mathbf{i},n) \quad , \quad k = 1,2$$

The implementation of functional transformations described above can be performed using photographic methods in the case of two-dimensional **i**. The result of learning in this case are photomasks realizing functions $a_n(\mathbf{i})$ that model the light flux $x(\mathbf{i},n)$, see Chap. 4, and coefficients a_0.

In the case of one-dimensional **i**, at the recognition of curves or electrical signals inside a fixed observation interval, functions $a_n(\mathbf{i})$and coefficients a_0 can be sufficiently and simply obtained by analogous facilities.

The sequential learning technique for the neuron with a feature continuum remains the same as in the case of the discrete feature set.

13.8 Application of the Adjustment Algorithm of the Multilayer Neural Networks with Flexible Structure for the Problem of Initial Condition Selection

Figure 13.17 shows the block diagram of the program realizing the process of sequential design for the piecewise linear divisional surface at the initial condition selection.

The idea of its application for the initial condition selection at the closed-cycle learning procedure of multilayer fixed structure neural networks is discussed below. The fixed structure of the multilayer neural network imposes constraints upon the number of neurons, at least in the first layer, and the algorithm can diverge. Therefore, the use of statistical methods for the calculation of the error recognition probability is necessary. However, this can be ignored at the initial condition selection procedure. There is a possibility to improve the divisional surface position after the algorithm termination by means of a sequential closed-cycle adjustment of each neuron with the learning quality test for the whole piecewise linear surface. The closed-cycle adjustment must be performed $2H_1$ times, i.e., $2H_1 \times M$ iterations, where H_1 is the number of the first-layer neurons, and M is the sample length. It is useful to decrease the sub-sample size up to some optimal value.

One of the methods to design such an optimal sub-sample is the deterministic selection of K (or $2K$, $3K$, etc.) patterns each belonging to one of the K pattern classes. In the case of the ultimate decrease of the sample size, one transfers to the deterministic or random selection of initial conditions in dependence of respective presence or absence of a priori information.

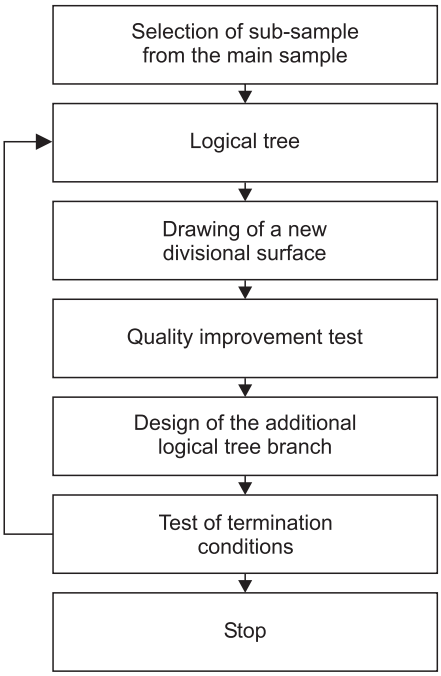

Selection of sub-sample from the main sample
Logical tree
Drawing of a new divisional surface
Quality improvement test
Design of the additional logical tree branch
Test of termination conditions
Stop

Fig. 13.17.
Block diagram of the program realizing the algorithm with flexible structure

13.9 About the Self-Learning Algorithm for Multilayer Neural Networks with Flexible Structure

The described technique for the adjustment of multilayer neural networks with flexible structure can be used to solve the problem of self-learning (clusterization) when the random sample with multi-modal distribution without instruction for patterns belonging to a particular class is present at the neural network input. Then the multilayer neural network is trained to recognize two pattern classes:

- The first class represents the initial sample;
- The second class represents an artificially generated random sample with the uniform probability distribution function in the range of feature variation. The feature space dimension for the samples of the first and second classes are equal.

Literature

[13-1] Galushkin AI (1970) Multilayer pattern recognition systems. Moscow, MIEM
[13-2] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energiya
[13-3] Gratchev LV, Simorov SN (1992) Statistical investigations of multilayer neural networks with. Neurocomputer 2:5–8
[13-4] Gerasimova AV, Gratchev LV (1992) To the problem of the learning sample representativity for the paradigm of the neural networks with flexible structure. Neurocomputer 3,4:3–6
[13-5] Dertuzos M (1967) Threshold logic. M, Mir, 343 p

Chapter 14

Informative Feature Selection in Multilayer Neural Networks

14.1 Statement of the Informative Feature Selection Problem in the Learning Mode

The problem of the informative feature selection is an independent problem in pattern recognition theory, and it has not yet been solved up to now. The existence approaches to this solution and description of so-called structural methods based on the multilayer neural network pattern recognition systems synthesis [14-1, 14-2] are discussed in this book.

Three statements form the basis of the proposed method:

1. The usual idea about the possibility of a preliminary informative feature selection before the stage of multilayer neural network adjustment is incorrect because the trained multilayer neural network already presents, explicitly or implicitly, in any known selection procedure;
2. Only the primary optimization criterion accepted for the given system can serve as a criterion of feature informativity. Any other criteria usually introduce additional errors and restrict their domain of applicability;
3. It is necessary to select multilayer neural networks of such types that are the most objective in the informative feature selection procedure, i.e., that provide the optimal solution in the sufficiently wide variation range of the multilayer neural network input signal characteristics (number of classes, distribution complexity inside the classes).

The problem of the informative feature selection was initially stated as the problem of selection of $N_1 = \text{const.}$ features out of N initial features. These N_1 selected features are supposed to provide minimum recognition probability error. This problem statement can be interpreted in another form: selection of minimum feature number N_1

P_0	P_1	P_2
0.95	0.9	0.85

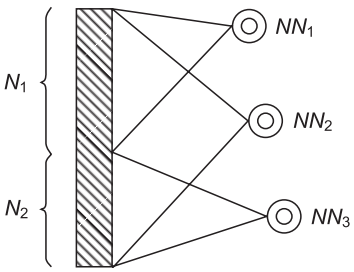


Fig. 14.1.
Selection of informative features in the initial feature space

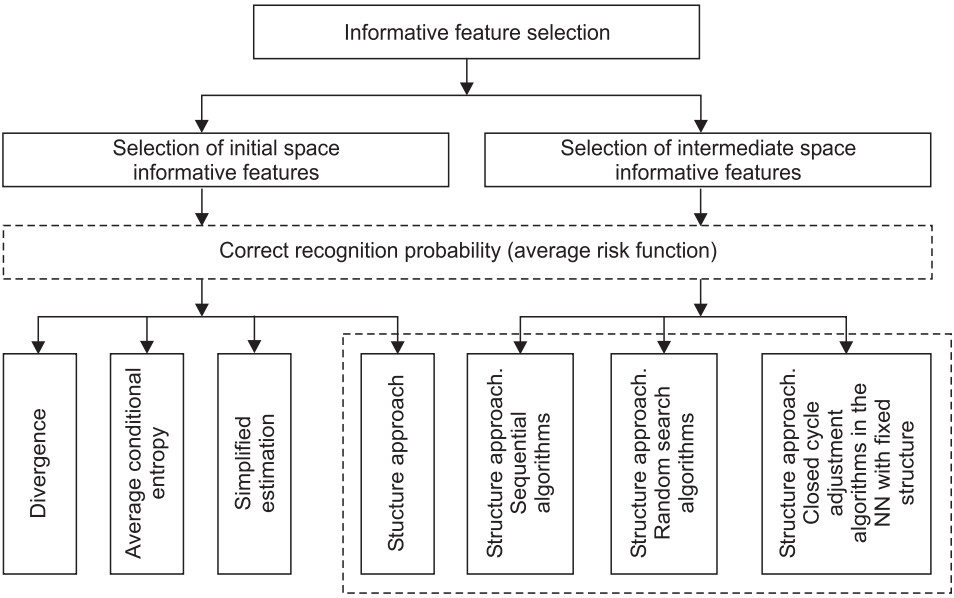

Informative feature selection
Selection of initial space informative features
Selection of intermediate space informative features
Correct recognition probability (average risk function)
Divergence
Average conditional entropy
Simplified estimation
Stucture approach
Structure approach. Sequential algorithms
Structure approach. Random search algorithms
Structure approach. Closed cycle adjustment algorithms in the NN with fixed structure

Fig. 14.2. Classification of methods for the informative feature selection

providing a given probability of correct recognition. Let us determine in this case the feature informativity criterion. Suppose that NN_0, NN_1 and NN_2 (Neural Network) with corresponding numbers of features $N = N_1 + N_2$, N_1 and N_2 (Fig. 14.1) provide the probabilities of correct recognition P, P_1 and P_2. If $P_1 > P_2$ then the group of N_1 features is more informative than the group of N_2 features. In this case, if the increment $\Delta P = P - P_1$ of the correct recognition probability is sufficiently enough to cover the expenses related to the system complexity increase due to the addition of N_2 features, then the use of the group of N_2 features is useful.

Such a problem statement for the informative feature selection is used in a wide range of practical tasks. For example, in some particular task of feature informativity estimation; the analysis of correct recognition probability P_{corr} can be performed for four feature groups: $(x_1, \ldots, x_N)$, $((x_1, \ldots, x_N) \cap x_i)$, $((x_1, \ldots, x_N) \cap x_j)$, and $((x_1, \ldots, x_N) \cap (x_i, x_j))$. We considered that such a particular problem as the selection of N_1^* features out of N ones for the achievement of maximum correct recognition probability cannot be solved without a solution in the statements described above.

In the particular case of a multilayer neural network with full connections, the problem consists in the minimization of the number of threshold elements in each layer. Further, the minimization criterion described above is also valid. Both aforementioned statements for the problem of informative feature selection are combined in the general structural approach to the this problem when the first layer is considered to be a priori organized in the form shown in Fig. 14.1.

Figure 14.2 shows the block diagram illustrating methods of informative feature selection in connection with the considered above problem statements and informative feature selection criteria.

This block diagram represents only the main ways of the considered solution, and it does not pretend to be complete, but it is only aimed at the introduction of the structural methods for the informative feature selection. The main development relates to divergence, conditional entropy, and some of their simplified estimations. They also include the approaches based on the component analysis and analysis of variance.

The main goal of the present chapter is the consideration of structural methods for the informative feature selection. They are based on the feature informativity estimation by the results of the multilayer neural network adjustment. At the solution of the adjusted multilayer neural network structure minimization, the minimization method depends on the adjustment method.

14.2 About Structural Methods for the Informative Feature Selection in the Multilayer Neural Networks with Fixed Structure

The structural methods of the informative feature selection are based on the initial feature space informativity estimation by parameters and structure of the optimally adjusted multilayer neural network. This section illustrates the structural methods for the informativity estimation in the example of a neuron.

Let us consider the multilayer neural networks of one neuron type or a neuron with a layer of nonlinear or random-nonlinear transformations (Chap. 1 and 2). The multilayer neural network in the form of one neuron is optimal for the pattern assemblages with normal multidimensional distribution laws and equal covariance matrices. In the case of the unit (to an accuracy of constant multiplier) covariance matrices, the level of class intersection by each of the features is determined by the corresponding inclination of the optimal linear divisional surface (Fig. 14.3).

The circles in Fig. 14.3 indicate the isolines of the densities $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$. If the main feature informativity criterion is the correct recognition probability as it was considered above, then it is easy to show that the i-th coefficient of the optimal divisional surface can serve as the relative estimation of the i-th feature informativity.

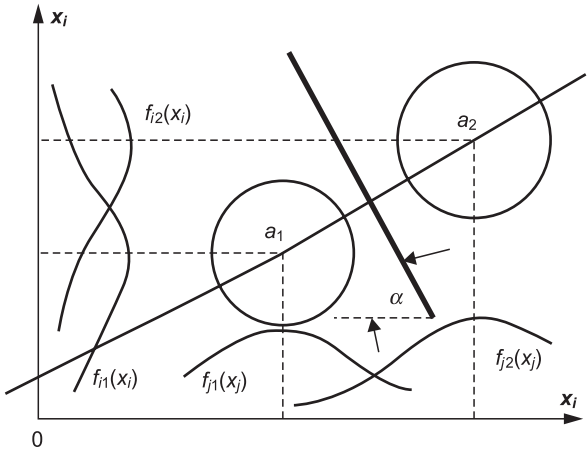


Fig. 14.3.
The proof of the possibility to use the neuron coefficients as the feature informativity estimations

The coefficients of the optimal neuron can also serve as the estimation of the feature informativity in the case of abnormal distributions. But it can be done only on the level of such an open-loop structure as a neuron. In the case of abnormal distributions and a nonlinear multilayer neural network, the neuron coefficients in the optimal nonlinear network represent the complex feature informativity estimations determined by the nonlinear transformation layer. A similar conclusion can be made for the three-layer Rosenblatt perceptron.

The structure minimization at the multilayer fixed structure neural network adjustment algorithm and the set of adjustment stages with random selection of initial conditions represent a separate problem. It includes the necessity to average the results of the adjustment procedure across the set of random initial condition injections. These injections are required for the local optimal adjustable coefficient search. The comparison of the minimized structures and the local optimal average risk function provides a direct rule for minimization of the number of neurons in the fixed structure multilayer neural network adjusted by the closed cycle.

It is necessary to consider distinctly the problem of minimization of the neuron number in the independent learning procedure for each neuron with the random initial condition selection separate for each neuron. After the independent training termination for H_1 neurons of the first layer, and, as a result, search of the local optimization functional extremum, the problem of selection by the adjustment results for one of the H_1 neurons that provides the optimization functional extremum value becomes trivial. The problem of selection of $H_1^0 < H_1$ neurons that provide the optimization functional extremum is rather complicated and maybe unsolvable in such a statement. This is illustrated by the example shown in Fig. 14.4. Here the error probability value is indicated in percents for each threshold selection. The numeric characters near the arrows indicate the class number.

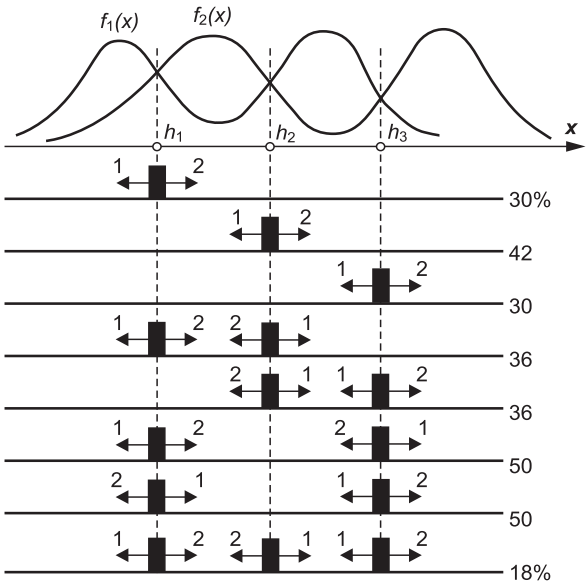


Fig. 14.4.
Example of minimization of the first-layer neuron number in the multilayer neural network: *1* – the first class; *2* – the second class

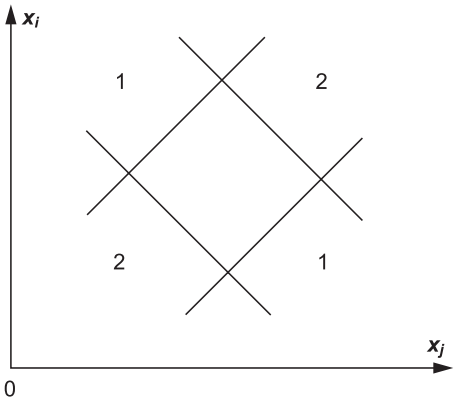


Fig. 14.5.
Illustration of the local optimality property of the informative feature selection procedure: *1* – the first class; *2* – the second class

This approach reveals its limitations in its generalization for the case of unknown and complicated form of distribution $f'(\mathbf{x}/\varepsilon)$. However, this limitation is completely explained taking into account the impossibility to select the informative features before the adjustment stage termination. Figure 14.5 illustrates this property in the particular example. The isolines of $f'(\mathbf{x}/\varepsilon)$ in the multi-modal case and four positions of the piecewise linear divisional surface providing the local extremum P_{corr} are shown here. Consequently, any informativity estimation at the fixed structure of the open-loop multilayer neural network is not only subjective but is also local because the adjusted multilayer neural network with fixed structure provides only a local optimization functional extremum. These arguments are also valid in the case of the self-learning mode.

14.3
Selection of the Initial Space Informative Features Using Multilayer Neural Networks with Sequential Algorithms of the First-Layer Neuron Adjustment

The main problem consists in the possibility to estimate the relative value of the correct recognition probability by the trained neural network structure form and the results of learning. Two feature groups are compared. Several methods of feature informativity can be proposed.

1. Let us assume that the given value of $P_{\text{corr}} = \text{const.}$ is provided (in particular, $P_{\text{corr}} = 1$) using multilayer neural networks with sequential learning algorithms of the first-layer neurons and some finite learning sample. Then, if the first neural network with characteristics $N_1, P_{1\text{corr}}$ has a higher number of neurons than the second neural network with characteristics $N_2, P_{2\text{corr}} = P_{1\text{corr}}$,then the group with N_1 features is less informative as compared with the group of N_2 features. This method of estimation is valid only under some conditions described below.
2. Let us assume that the minimal recognition error is provided at each step of the first layer learning procedure. The learning results are represented in Fig. 14.6a by the curves of P_{corr} vs. the number of the first-layer neurons H_1 on the feature samples N_1 (NN_1) and N_2 (NN_2). It is seen that feature group N_1 is less informative than N_2. This method includes that which is described in p. 1 as its particular case.

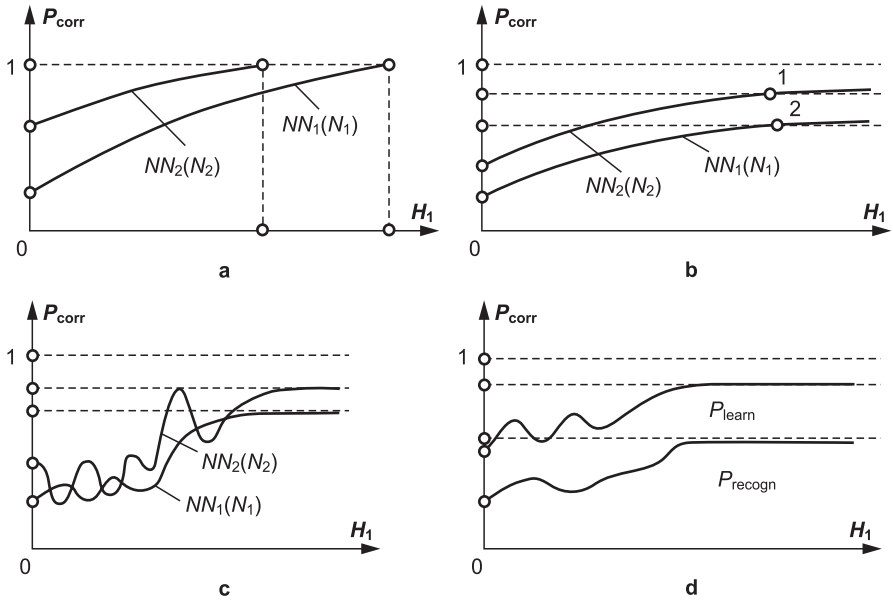


Fig. 14.6. Selection of informative features using a multilayer neural network with flexible structure

3. Dependence $P_{corr}(H_1)$ has the form shown in Fig. 14.6b when the learning sample is sufficiently large. The curve $P_{corr}(H_1)$ is close to its asymptote. This means the transfer from the statistical mode to the deterministic one. The informativity estimation is reduced to the comparison of the steady values of $P_{corr}(H_1)$.
4. Fig. 14.6b,c shows the general case of a non-optimal adjustment algorithm for the first-layer of neurons of the multilayer neural network. The informativity estimation in this case is performed either according to p. 3 or at any H_1 but with the remark that this estimation is valid for the given adjustment algorithm and the given number of the first-layer neurons.
5. It was assumed above that the learning sample is the same as the initial sample. In order to take into account the case when the representation of the learning sample in the initial sample is smaller, one must perform the learning procedure on some part ΔM_i of the initial sample. The recognition by the trained multilayer neural network is carried out on the full sample. The analysis of the learning $P_{learn}(H_1, \Delta M_i)$ and recognition $P_{recogn}(H_1)$ results that are illustrated by Fig. 14.6d allow one to estimate the stationary properties and representativity of the learning sample, as well as informativity of different feature groups.

14.4 Neuron Number Minimization

The adjustment sequence process for the first-layer neurons of the multilayer neural network (Chap. 13) is described by the graph in the form of the logical tree. Each top of the tree corresponds to a neuron and an increment P_{corr} that takes place after the

introduction of this neuron. Such a graph represents the initial information for the aforementioned minimization procedure. This graph can be minimized according to one of the following statements: to minimize the number of tops under the given P_{corr}, or to provide maximal P_{corr} under the given number of tops.

Figure 14.7 represents an illustration of the initial information for the graph minimization. The top number of the initial graph is indicated at the left part of the circle. The top number of the resultant minimized graph is indicated at the right part of the circle. The number of each graph rib coincides with the number of the divided region (Chap. 13). The sub-region containing the maximum number of vectors of the first and second classes is selected for the next division. The dashed lines correspond to the sub-regions with a relatively small number of vectors. The corresponding increment P_{corr} (either positive or negative) is written near each graph top in the square brackets. The logical tree optimization is performed in the following way:

1. The increments P_{corr} are compared in the case of the first branching (neurons 3 and 8 in the initial graph). The neuron with maximum P_{corr} is selected for the optimized graph (neuron 3 in this case);
2. Then the neurons of the given and the next branchings are compared by ΔP_{corr} (neurons 8 and 4) and again the neuron with maximum P_{corr} is included into the optimized graph;
3. The process continues until the sum of correct recognition probability achieves some given value P_{corr} or the number of tops achieves some given value.

The described procedure results in the optimal tree traversal as it is shown in Fig. 14.7a in the circles (tops) from the right. Figure 14.7b presents the result of optimization of the graph in Fig. 14.7a for two criteria: $P_{corr} > 0.7$ and $P_{corr} > 0.73$. The course of tree traversal in the optimal graphic design does not coincide with that on the learning stage.

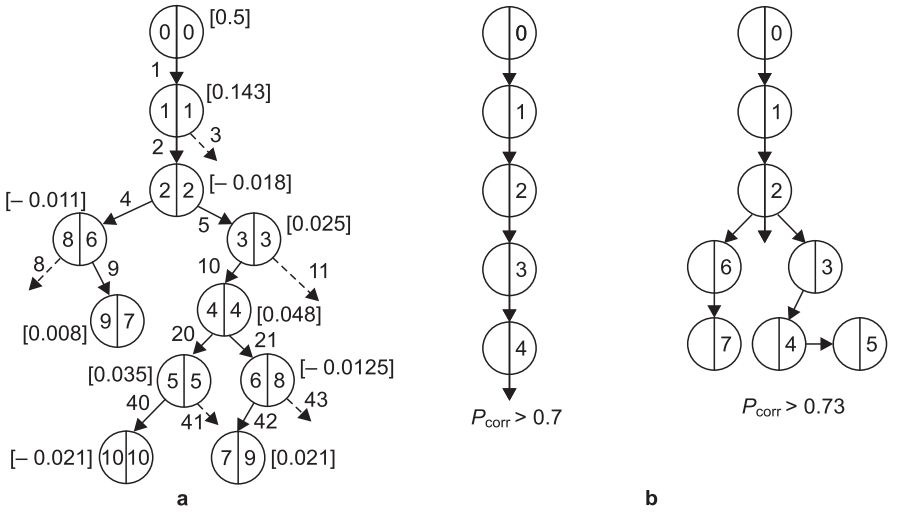


Fig. 14.7. Minimization of the first-layer neurons of a multilayer neural network with flexible structure

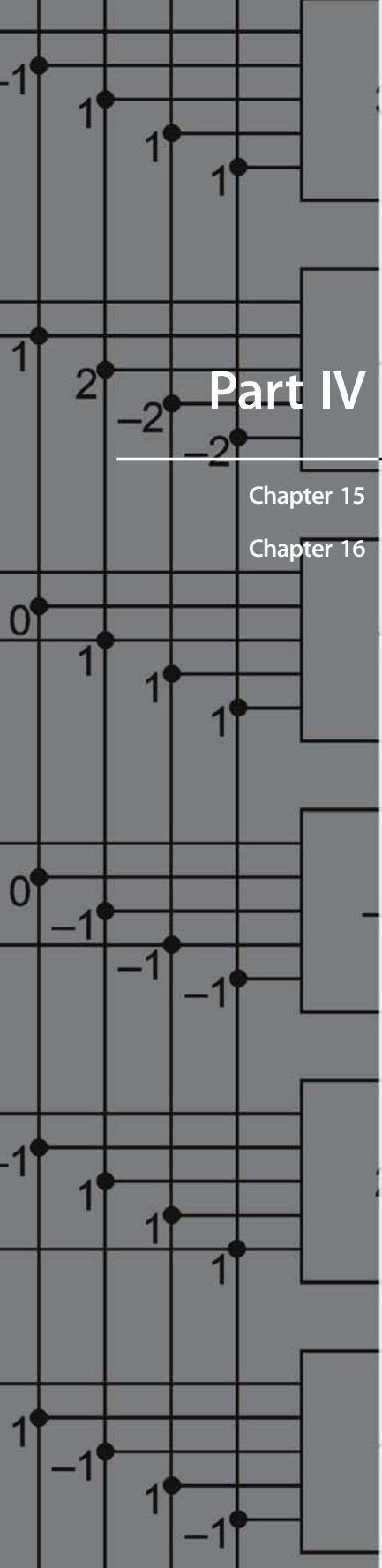
The idea of the use of sequential algorithms for the second-layer neuron learning consists in the use of sequential algorithms for each learning vector with consideration of the weight determined by the P_{err} in the sub-region corresponding to this vector. The principle of the number minimization for the second-layer neurons as well as for the neurons of the following layers is the same in this case as for the first-layer neurons. Notice that the significance of the neuron number minimization decreases with the increase of the layer number due to the specific properties of the open-loop multilayer neural network structure (decrease of the neuron number due to the data compression from the first layer to the output).

14.5 About the Informative Feature Selection for Multilayer Neural Networks in the Self-Learning Mode

All the problem statements for the informative feature selection described in Sect. 14.1 are valid in the self-learning mode. Only the informative feature selection criterion is modified. In the learning mode, such a criterion is the value of the average risk function (in particular, the correct recognition probability) whereas in the self-learning mode this criterion is the value of the special average risk function. The methods of the informative feature selection described in Sect. 14.3 concerning the learning mode and recognition systems with flexible structure, as well as concerning corresponding methods of the network structure minimization can be methodologically generalized in a relatively simple manner for the case of the self-learning mode. The structure minimization for the recognition system with fixed structure must be performed by means of the analysis of the adjusted multilayer neural network structure as well as the analysis of the obtained value of the special average risk function.

Literature

[14-1] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow, Energiya, p 367
[14-2] Galushkin AI (1996) Summary and perspectives of the multilayer neural network theory development (1965–1995) in the proceedings of the Neurocomputer scientific center. Moscow



Part IV Neural Network Reliability and Diagnostics*

Chapter 15 Neural Network Reliability

Chapter 16 Neural Network Diagnostics

* This part is written in collaboration with Fomin Yu. I.

Chapter 15

Neural Network Reliability

15.1 Methods for the Neural Network Functional Reliability Investigation

The first attempts to estimate the neural network functional reliability were experimental [15-1] or qualitative [15-2]. The qualitative estimations showed that the neuron-like elements are characterized by the logical redundancy [15-2, 15-3], i.e., the failures of some elements do not result in the errors at the neural network output.

The attempts of the analytical neural network reliability investigation face mathematical difficulties. Some works [15-4 to 15-6] claim the impossibility of the full analytical neural network investigation. Several particular neural networks were considered in these studies and analyzed on the basis of the Markovian process theory by means of graphic design. The disadvantage of such an approach is related to the presence of the system of differential equations that is very complicated for the explicit solution even in the case of the simplest graphs.

The work [15-7] deals with the neural network reliability in the sense of its logical stability. The logical stability is investigated with the help of stability maps that can be written in the explicit form only for the simplest neural networks such as the threshold element triplet. This approach cannot be used in practice.

In the study [15-7], some empirical expressions for the specific neural networks with several constraints on their complexity were derived. But the calculations performed according to these expressions cannot simulate the objective estimation of the reliability functional for the considered threshold element networks.

The attempt to investigate the reliability of one neuron taking into account that its weighting coefficients and input values are random is performed in [15-9 to 15-12]. The authors failed to obtain an analytical result even in the simplest considered case. These studies are characterized by several disadvantages: all the analytical calculations were based on the experimental data and extrapolated neuron probabilistic relay function (PRF); some intermediate results, for example, the PRF mathematical expectation in [15-7], were obtained by means of additional simplifications because of their mathematical complexity; the final expression can be integrated only using numerical methods. However, we consider this investigation as the most successful analytical investigation of the neuron reliability because it takes into account the neuron functional structure and the probabilistic model of its functioning.

Thus, one can conclude that analytical investigation of the multilayer neural network reliability must be based on the principally new approach, or this reliability must be studied experimentally by means of Monte Carlo methods.

The reliability of the so-called generalized threshold element is analytically investigated in [15-13]. The originality of such an approach relates to the fact that any combinational network of neurons (multilayer neural network with sequential or cross [15-14] connection) can be realized in the form of a neuron layer. Since the functional reliability of the neuron layer can sometimes be reduced to the reliability of one neuron, then the analysis investigation in this case is principally significant.

The experimental investigation of reliability can be performed on three levels: circuit, functional, and logical. Functional and logical levels are considered in [15-4, 15-5]. The neuron failures are usually divided [15-15] into two classes: parametrical and catastrophic failures. The parametrical failures are caused by the gradual changes of weighting coefficients and threshold under the influence of the external factors (supply voltage or temperature changes, components aging, etc.). The catastrophic failures are caused by disconnection faults or short-circuit failures.

Experimental methods of the parametrical failure investigation are based on the assumption that the weighting coefficients and thresholds of all the neurons are the random values with normal probability distributions. The distribution parameters are considered to be known and to be modeled by the Monte Carlo method. Such an investigation technique allows one not only to analyze a wide class of multilayer neural networks and make resumptive conclusions, but also to estimate the parametrical reliability of particular implementations.

It is clear that the multilayer neural network's functional reliability investigation is not complete without catastrophic failure analysis. A special technique for such analysis was developed and used to study failures of the logical constant type at the neuron input-output (input-output stuck-at faults). The analysis is based on the successive modeling of stuck-at faults of all types for each neuron and the following calculation of the failure-free performance probability. Such an approach allows one to reveal "potentially dangerous" failures that result in the drastic decrease of the failure-free performance as compared with the other failures. The obtained results can be used for eliminating the possibility of "potentially dangerous" failures in the design planning stage.

The developed experimental technique for reliability investigations allows one to obtain some quantitative characteristics of multilayer neural network reliability and provides the practical possibility to perform reliability investigations of concrete neural network implementations.

15.2 Investigation of Functional Reliability of Restoring Organs Implemented in the Form of Multilayer Neural Networks

The problem of reliability of digital devices implemented on any basis and particularly on the basis of multilayer neural networks is of great interest at this time due to the complication of computers and functions performed on their base. This problem is of special significance for computers functioning in a nonrestorable mode (for example, on-board computers), i.e., without access to reparation performance.

After the foundational work of von Neumann [15-16], the synthesis of reliable digital devices made of unreliable elements was investigated by different researchers. The works [15-17 to 15-21] must be specially mentioned in this connection. The proposed

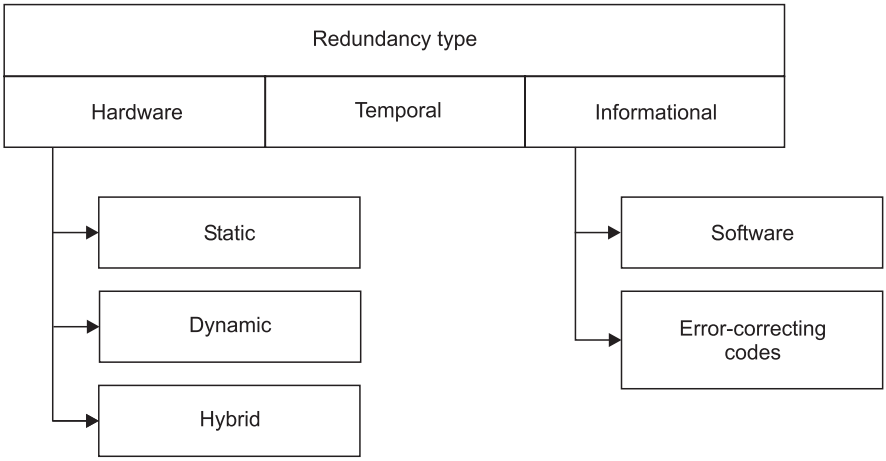


Fig. 15.1. Classification of methods for redundancy introduction

methods are based on the introduction of some logical redundancy into the digital device construction. The redundancy can be classified [15-22] as hardware (structural, [15-23]), temporal and informational. Such a classification is conditional because the redundancy of any of these types is usually accompanied by redundancies of other types. The classification represented in Fig. 15.1 is carried out according to the redundancy properties to increase the system reliability.

The hardware redundancy can be used at any functional level beginning from components up to the whole system. Three types of hardware redundancy can be implemented according to the activity of the main and redundant components [15-24]: static [15-25], dynamic [15-4], and hybrid [15-2, 15-26, 15-27].

In the case of static redundancy, all the components, main and redundant, are functioning. Overcoming the failure effects is performed automatically by the error correction at the expense of redundancy in the system components.

In the case of dynamic redundancy, the redundant devices start functioning only under the requirement to substitute the failure units. This redundancy provides the system self-restoration. It requires the use of testing and diagnostic methods for failure detection.

The hybrid redundancy is the combination of static and dynamic redundancies. Some duplicated devices are permanently functioning. The failure of one of them results in its substitution by the redundant device.

The following main peculiarities of the static redundancy are usually distinguished:

- Error correction without interruption of functioning;
- Correction of errors that occurs as a result of permanent failures as well as short-duration failures;
- Significant increase of the failure-free performance probability of the low-reliable devices at the expense of the low redundancy level;
- Significant advantages of the static redundancy are the universality and the absence of necessity to develop special software for detection, localization and correction of errors.

The scheme of the majority redundancy [15-17, 15-28] is often used for the static redundancy design. It implies the *n*-fold duplication of components or units, and the outputs of backup units are loaded by the restoring organ [15-17, 15-29]. The restoring organ realizes the following decision rule in the case of majority voting: its output is equal to the value that is accepted by the majority of the restoring organ inputs.

The dependence of the restoring organ free-failure functioning probability upon the type of decision rule is investigated. Different restoring organ schemes implemented in the form of multilayer neural networks are considered. Accessible regions of parameter variation for the restoration of organ optimal functioning are discussed.

15.3 Investigation of Multilayer Neural Network's Functional Reliability

Methods of investigation of multilayer neural network's functional reliability can be classified as analytical and experimental ones. Analytical investigation at the level of the neuron deals with mathematical complexity. Consequently, the main attention is paid to the experimental investigation of multilayer neural network reliability (see Sect. 15.4 below). The main results described below are represented in [15-30, 15-33 to 15-36].

The multilayer neural network class with binary inputs is considered. The investigation of multilayer neural network's functional reliability is based on the following settings:

a Functional reliability criterion;
b Probabilistic model of the neural network functioning;
c Set of the input values.

The correct multilayer neural network functioning probability and the output signal probability distribution function are considered below as a functional reliability criterion.

The probabilistic model of the neural network functioning depends on the physical essence of the considered failure types (parametrical, catastrophic). For example, in Sect. 15.4, parametric failures are considered, and the weighting coefficients and thresholds of all the neural network elements are assumed to be random.

It is useful to divide the experimental investigation into several stages according to the number of neuron failure classes. The neuron failures are usually divided into two classes [15-15, 15-37]: parametric and catastrophic ones. The experimental investigation is therefore divided into two stages: the investigation of reliability for parametrical failures (parametrical reliability) and the investigation of catastrophic failures (catastrophic reliability).

The experimental technique was developed for the class of neural networks with sequential connections. It allows one to analyze the neural networks with an arbitrary number of inputs, arbitrary number of neurons in the layers, and arbitrary number of layers.

The parametrical failures of neurons [15-15] include the errors at the neuron outputs caused by the gradual changes of weighting coefficients and threshold under the influence of the external physical factors: temperature changes, supply voltage changes, etc.

Experimental methods of the parametrical failure investigation using the Monte Carlo method include the normal probability distribution modeling for weighting coefficients and thresholds.

The catastrophic failures include the failures caused by disconnection faults or short-circuit failures. The failures of such a type can be reduced to the failures of the logical constant type (const. = 0 and const. = 1) at the neuron input-output [15-37]. It is assumed that the neuron failures are random, independent, and equally probable. The deterministic choice of the failure type and the number of the failure neuron is performed at the catastrophic reliability investigation. The considered approach allows one to reveal "potentially dangerous" failures that result in the drastic decrease (with respect to some a priori given value) of the failure-free performance.

The obtained concrete values of the correct recognition probabilities can be used for eliminating the possibility of "potentially dangerous" failures in the design planning stage. The advantage of the developed experimental technique for reliability investigations consists in the fact that it allows one not only to analyze a wide class of multilayer neural networks and to make resumptive conclusions, but also to estimate the reliability of particular implementations.

15.4 Investigation of the Neural Network's Parametrical Reliability

Several logical function implementations and two- and three- layer neural networks with a different number of neurons in the first layer were considered in the investigation of the neural network's parametrical reliability. The following objectives were pursued in the experimental study:

1. To analyze parametrical reliability, i.e., the correct probability dependence on variance $D[a]$ of the weighting coefficients and thresholds, at different fixed shifts of mathematical expectation $-\Delta a$ for different one-neuron realizations of some logical function; to find an optimal realization on the base of this analysis;
2. To analyze the change of correct recognition probability with the change of variance of the weighting coefficients and thresholds with dependence on:

a The increase of the first-layer neurons in the two-layer neural networks;
b The increase of the first-layer neurons in the three-layer neural networks;
c Transfer from the two-layer to the three-layer neural network realizing the same logical function and fixed number of the first-layer neurons;
d Transfer from the two-layer to the three-layer neural network with the same total number of neurons.

The stages of the performed investigations are described below.

Stage 1. The optimal realization selection by the maximum parametrical reliability criterion is shown in the example of three different majority votes. Each neuron realizes some hyperplane crossing a unit N-dimensional hypercube (N is the number of neuron inputs) and separating two vertex classes: (1) vertexes with the unit component number less than $N/2$ and (2) vertexes with the unit component number more than $N/2$ (the zero component number is less than $N/2$). Let us choose a neuron with unit weighting coefficients out of all of the neuron set realizing the majority decision

rule. Their corresponding hyperplanes are parallel and cross the coordinate axes under 45°. The majority element that determines the majority rule is usually chosen in this case, and it satisfies the following equation:

$$y = \operatorname{sign}\left(\sum_{i=1}^{N} x_i - a_0\right) \tag{15.1}$$

where $x_i \in \{0,1\}$ are the input values and $a_0 = (N-1)/2$ is the threshold of the majority element

$$\operatorname{sign}(x) = \begin{cases} 0 \ , & x \le 0 \\ 1 \ , & x > 0 \end{cases} \tag{15.2}$$

This realization is the limiting case for the threshold decrease because (15.2) is not valid at

$$a_0' = \frac{N-1}{2} - \varepsilon$$

where ε is an indefinitely small value. The hyperplane corresponding to the majority element for the case $n = 3$ is shown in Fig. 15.2.

Here and below, the criss-crosses indicate the input and intermediate values corresponding to the unit output, and the circles indicate the values corresponding to the zero output. A realization limiting the threshold increase of the considered family of neurons is a neuron with the threshold $(N+1)/2$ described by the expressions (15.1) and (15.2).

Figure 15.3 shows a hyperplane corresponding to such a realization for $N = 3$.

Therefore, let us investigate neurons with unit weights and thresholds from the interval $[(N-1)/2, (N+1)/2]$. Three neurons with the thresholds $(N-1)/2$, $(N+1)/2$, and $N/2$ were taken for the experimental study.

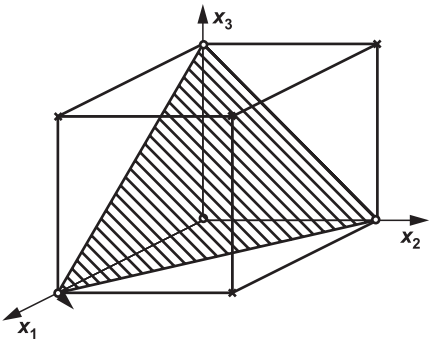


Fig. 15.2. Hyperplane realized by the majority element in the space of inputs ($a_0' = (N-1)/2$)

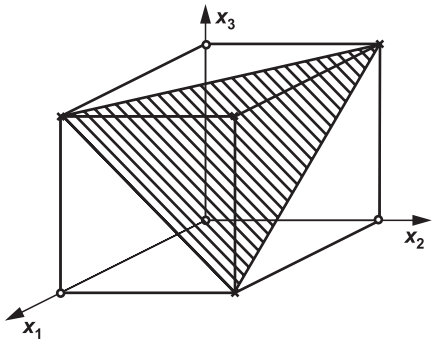


Fig. 15.3. Hyperplane realized by the neuron with unit weights and the threshold (N + 1)/2 in the space of inputs

Since the change of the weighting coefficient mathematical expectation shift $-\Delta a$ indicates some hyperplane shift and rotation, then the investigation of the three aforementioned realizations for different Δa results in the determination of the optimal realization for the arbitrary Δa. The experiment was performed for three values of the mathematical expectation shifts: $\Delta a = 0.15; 0.0; -0.15$. The average experimental curves are represented in Fig. 15.4a–c.

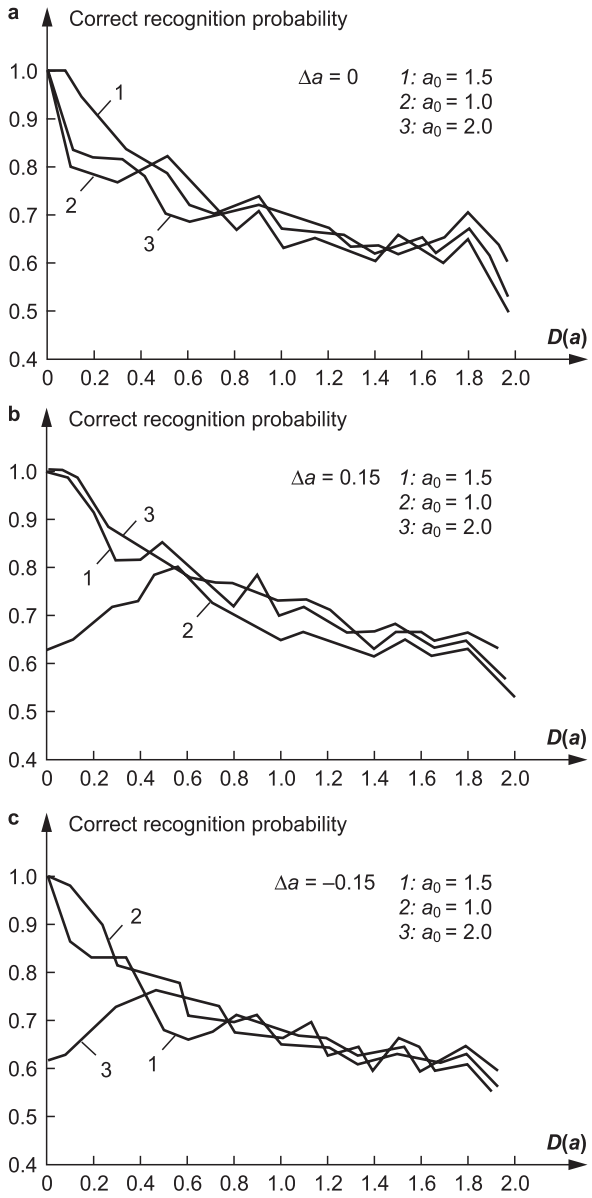


Fig. 15.4.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of three realizations of one neuron: **a** $\Delta a = 0$; **b** $\Delta a = 0.15$; **c** $\Delta a = -0.15$

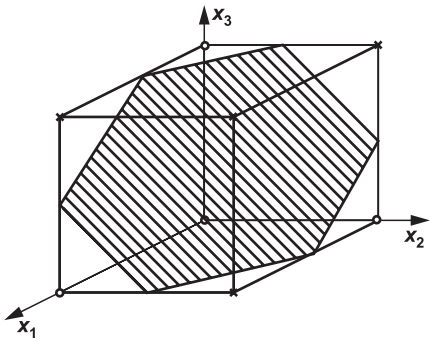


Fig. 15.5.
Hyperplane realized by the optimal neuron

Variance $D[a]$ is measured in the same units as for the weighting coefficients. It is seen that the correct recognition probability decreases in general with the variance increase. In Fig. 15.4a, at $\Delta a = 0$, the optimal realization is the realization of the neuron with the threshold $a_0 = 1.5$. In Fig. 15.4b, at $\Delta a = 0.15$, the optimal realization is the realization of the neuron with the threshold $a_0 = 2$. In Fig. 15.4c, at $\Delta a = -0.15$, the optimal realization is the realization of the neuron with the threshold $a_0 = 1.0$. The averaging of all corresponding correct recognition probability values across realizations for three values of Δa shows that the neuron with the threshold $a_0 = 1.5$ represents an optimal realization for arbitrary Δa. Thus, one can make a general conclusion for the case of the optimal neuron with the threshold $a_0 = N/2$, and, consequently, with the thresholds $(N - 1)/2$ at $\Delta a > 0$ and $(N + 1)/2$ at $\Delta a < 0$.

The value Δa for the real neurons can be either positive or negative. The optimal neuron realization in this case corresponds to the hyperplane equidistant from the symmetrical points of both classes, i.e., the hyperplane drawn through the middles of the corresponding hypercube ribs. This hyperplane position for $N = 3$ is represented in Fig. 15.5.

The following conclusion can be made on the basis of the obtained results. Each hyperplane must be drawn through the middles of the corresponding hypercube ribs at the multilayer neural network synthesis on the binary input signal set. Then the neurons and the neural network possess maximal parametrical reliability with respect to all other possible realizations. Only such optimal neural networks with optimal neurons are considered at all the following stages.

Stage 2. Let us consider six different neural networks with two,three,…,seven neurons in the first layer. All the neurons realize the optimal hyperplanes according to the results obtained at stage 1.

Figure 15.6 shows one of such realizations (two-layer neural network with $H_1 = 2$ neurons in the first layer), where x_i^j is the output value of i-th neuron of j-th layer. The experimental curves for the correct recognition dependence on the variance at $\Delta a = 0$ are represented in Fig. 15.7a,b (curves for $H_1 = 2,4,6$ and $H_1 = 3,5,7$ are displayed in different figures for clarity).

The obtained results show that the parametrical reliability is constant when H_1 increases and variance is small: $0 \le D[a] \le D^*[a]$ $(D^*[a] \cong 0.6)$. The probability increases with the increase of H_1 in the case $D^*[a] \le D[a] \le 2$. This probability increase is especially stressed at $H_1 = 6.7$. Consequently, one can make a conclusion that two-layer neural

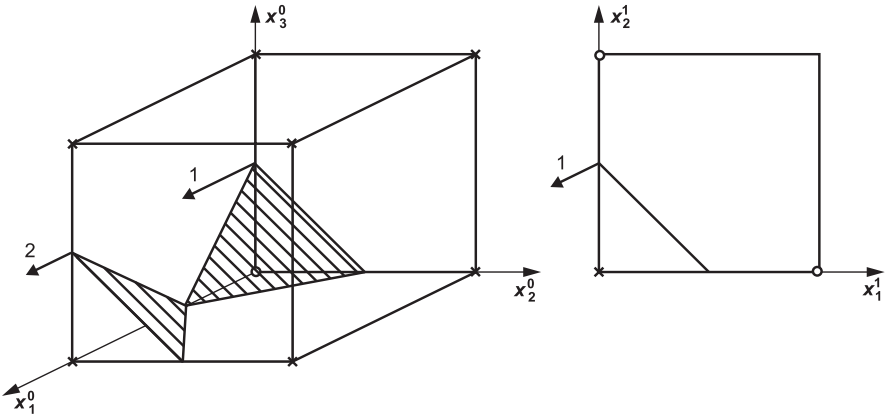
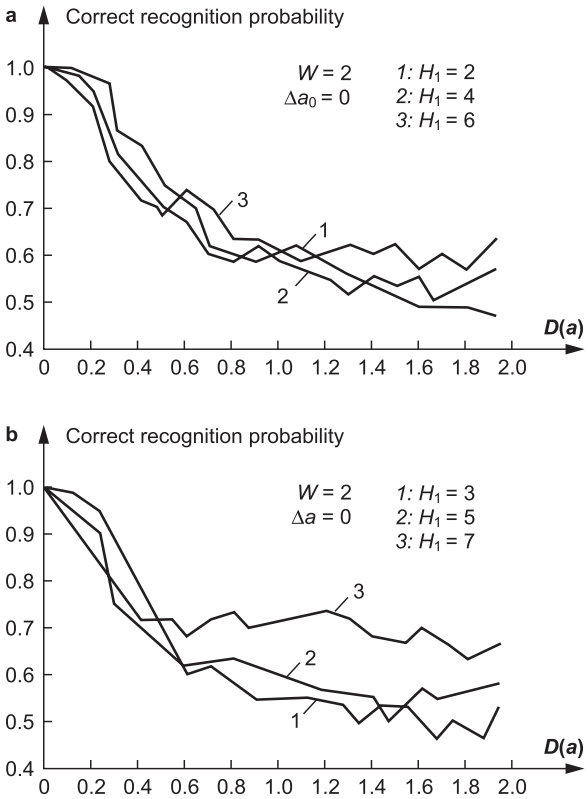


Fig. 15.6. Hyperplanes realized by the neurons of the two-layer neural network

Fig. 15.7.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of two-layer neural networks with a different number of neurons in the first layer



networks are characterized by the improvement of the correct recognition probability dependence upon the variance (the increase of the parametrical reliability) with the increase of H_1.

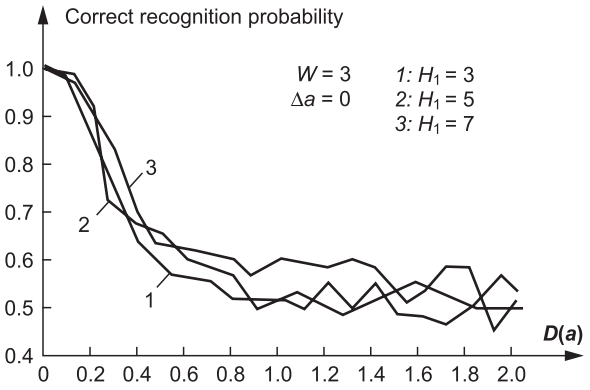


Fig. 15.8.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of three-layer neural networks with a different number of neurons in the first layer

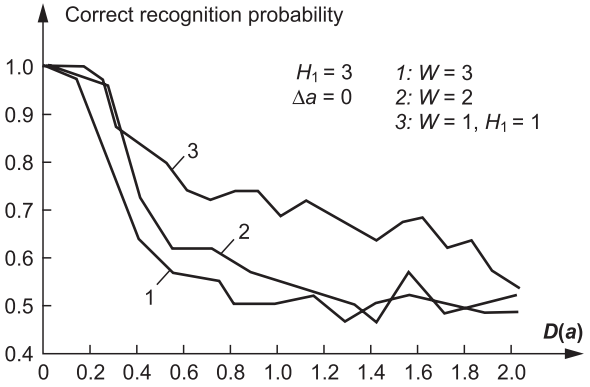


Fig. 15.9.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of two- and three- layer neural networks realizing the same logical function

Stage 3. Let us consider five different neural networks with three, four, …, seven neurons in the first layer, two neurons in the second layer, and one neuron in the third layer. All the neurons realize optimal hyperplanes. The parametrical reliability curves for $H_1 = 3,5,7$ are represented in Fig. 15.8.

On the basis of the obtained curve analysis, one can make a conclusion similar to that made at stage 2: three-layer neural networks are characterized by the increase of the parametrical reliability with the increase of H_1.

Stage 4. This investigation stage deals with the problem concerning the change of parametrical reliability at the transfer from the two-layer neural network to the three-layer neural network that realizes the same logical function under the fixed value of H_1. The previously obtained results at stages 2 and 3 can be used in this case. The corresponding comparative characteristics for two- and three- layer neural networks with $H_1 = 3, 5, 7$ are represented in Figs. 15.9–15.11.

It is seen that the transfer from the two-layer neural network to the three-layer neural network results in the decrease of the parametrical reliability. For demonstrativeness, in addition to the aforementioned curves, the curve for one neuron realizing the same logical function is represented in Fig. 15.9. One can make a conclusion that the increase of the number of layers results in the decrease of parametrical reliability.

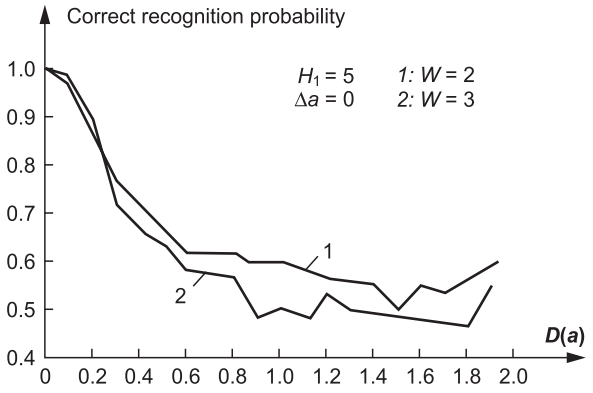


Fig. 15.10.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of two- and three- layer neural networks with the same number of neurons in the first layer ($H_1 = 5$)

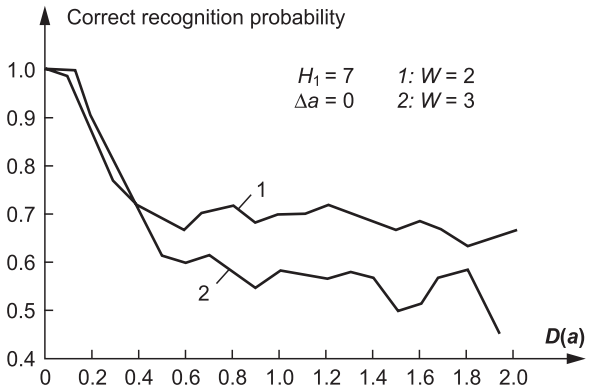


Fig. 15.11.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of two- and three- layer neural networks with the same number of neurons in the first layer ($H_1 = 7$)

Stage 5. Let us consider all possible two- and three-layer neural networks from the assemblage that was described above. The total number of neurons in these neural networks is assumed to be equal. There are only three such pairs: with six neurons (3 + 2 + 1 and 5 + 1), seven neurons (4 + 2 + 1 and 6 + 1), and eight neurons (5 + 2 + 1 and 7 + 1). The corresponding curves are represented in Fig. 15.12a–c. One can conclude that the two-layer neural network possesses the highest parametrical reliability.

The performed experiments allow one to make the following conclusions:

1. The correct recognition probability decreases with the increase of variance of the weighting coefficients and threshold at the fixed mathematical expectation shift;
2. The neuron possesses maximal parametrical reliability when the hyperplane realized by this neuron is drawn through the middles of the corresponding hypercube ribs;
3. Neural network parametrical reliability increases with the increase of the number of neurons in the first layer in the case of both two- and three-layer neural networks;
4. The transfer from the two-layer neural network to the three-layer neural network realizing the same logical function with the same number of neurons in the first layer results in the decrease of the parametrical reliability;

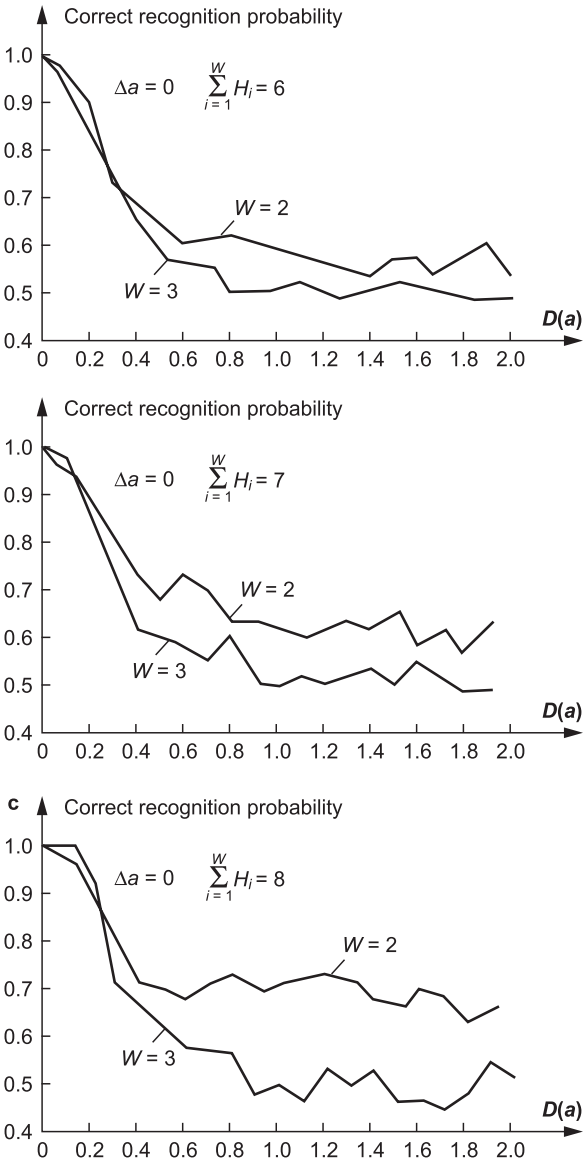


Fig. 15.12.
Average curves for correct recognition probability dependence upon the variance of weighting coefficients in the case of two- and three-layer neural networks with the same total number of neurons: **a** $H_1 = 6$; **b** $H_1 = 7$; **c** $H_1 = 8$

5. The comparison between two- and three-layer neural networks with the same total (across all the layers) number of neurons shows that the two-layer neural network possesses higher parametrical reliability.

The following plan of experiment can be performed in addition to that described above in the case of a sufficiently large number of the required multilayer neural network realizations.

To investigate the dependence of the neural network parametrical reliability upon

1. The number of layers at the fixed values of H_1, H_2,…;
2. Dimensionality of the input signal of the neural network H_1, W;
3. The number of neurons at the fixed number of the neural network inputs.

15.5 Investigation of the Multilayer Neural Network's Functional Reliability in the Case of Catastrophic Failures

The experimental methods for investigation of the multilayer neural network functional reliability in the case of catastrophic failures consist in the successive modeling of the single-fold failures of the logical constant type at the neuron inputs and outputs and calculation of the correct recognition probability value for each failure.

Table 15.1. Results of investigations of the neural network reliability

Number of		Failure type			CRP
layer	neuron				
1	1	Input	1	0	0.875
				1	0.875
			2	0	0.875
				1	0.875
			3	0	0.875
				1	0.875
		Output		0	0.875
				1	0.375
	2	Input	1	0	0.875
				1	0.875
			2	0	0.875
				1	0.875
			3	0	0.875
				1	0.875
		Output		0	0.875
				1	0.375
	3	Input	1	0	0.875
				1	0.875
			2	0	0.875
				1	0.875
			3	0	0.875
				1	0.875
		Output		0	0.875
				1	0.375
2	1	Input	1	0	0.875
				1	0.375
			2	0	1.000
				1	1.000
			3	0	0.875
				1	0.375
		Output		0	0.750
				1	0.375
	2	Input	1	0	1.000
				1	1.000
			2	0	0.875
				1	0.375
			3	0	1.000
				1	1.000
		Output		0	0.875
				1	0.375
3	1	Input	1	0	0.750
				1	0.375
			2	0	0.875
				1	0.375
		Output		0	0.625
				1	0.375

CRP – Correct Recognition Probability.

Let us describe the process of the "potentially dangerous" failure search on the basis of the network catastrophic reliability investigation and methods of the network logical redundancy determination.

The results of the investigation are represented in Table 15.1. They indicate that on the failure class of the logical constant type at the neuron's input-output, under the failure equal probability, the analyzed network possesses the logical redundancy coefficient of 6/46, i.e., at six failures out of 46 possible ones, the correct recognition probability amounts to 1. One can also select "potentially dangerous" failures taking the minimum acceptable value of the correct recognition probability. Let it be equal to 0.75. Then only eleven "potentially dangerous" failures having the correct recognition probability of 0.375 exist. For example, const. = 1 at the output 1 of the first layer neurons; const. = 1 at the input 1 of the second layer neurons, etc.

Thus, the experimental study of the neural network catastrophic reliability allows one to take into account and to use the logical redundancy for eliminating the possibility of "potentially dangerous" failures. This can significantly increase the reliability of the designed logical devices.

Literature

[15-1] Coates CL, Lewis PM (1964) DONUT-a threshold gate computer. IEEE Trans. ELEMENT. Comp.; No. 3, vol. EC-13

[15-2] Abbakumov IS, Chernyshv NA (1979) Majority redundancy systems with readjustable structure. Automatics and computer facilities 3:31–36

[15-3] Blyum M, Onesto M, Verbik L (1966) Acceptable neuron errors for neural network failure-free performance. In: Methods of redundancy introduction for computer systems. Pougatchev VS (ed) – Moscow: Sov. Radio, p 84–87

[15-4] Potapov VI, Palyanov IA (1972) To the functional reliability estimation of the redundancy readjustable homogeneous computer structure. In: Computer facilities in the in the systems of flight vehicle control, part II, vyp. 23, Moscow

[15-5] Potapov VI (1977) Analysis and synthesis of highly reliable digital and computer threshold unit logical structures. Novosibirsk, p 80

[15-6] Potapov VI (1968) Functional reliability of the formal neuron networks. Automatics and computer facilities 1:37–43

[15-7] Maitra KK (1966) Reliable automats synthesis and neuron circuit stability. Bionics, parts II and III. Kiev: KVIITRU, p 37–43

[15-8] Lopin VN (1975) About reliability of the controlled threshold element network under constraints upon its complexity. In: Adaptive control systems. Kiev, p 91–97

[15-9] Serapinas KL, Jukauskas KP (1969) Threshold element reliability (1. Probabilistic estimation of the relay part of the threshold element). Trudy AN LitCCP, ser. B, 2(57):159–162

[15-10] Jukauskas KP, Serapinas KL (1969) Threshold element reliability (2. External noise influence.) Trudy AN LitCCP, ser. B, 4(59):213–216

[15-11] Jukauskas KP, Serapinas KL (1970) Threshold element reliability (3. Generalized VRF for the group of threshold elements). Trudy AN LitCCP, ser. B, 3(62):153–157

[15-12] Jukauskas KP, Serapinas KL (1971) Threshold element reliability (5. Determination of the average threshold element error taking into account the input signals and weighting coefficient parameter spread). Trudy AN LitCCP, ser. B, 1(64):231–236

[15-13] Gill A. (1974) Linear sequential machines. Moscow: Mir, p. 287

[15-14] Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow: Energiya, 367 p

[15-15] Palianov IA, Potapov VI (1977) Failure diagnostics and synthesis of digital structures based on the threshold logical units. Novosibirsk, p 78

[15-16] Neumann J (1955) Probabilistic logic and synthesis of reliable organisms on the basis of unreliable components. In: Automats, Moscow: Foreign litr., p 68–138
[15-17] Pirs W (1968) Design of reliable computers. Moscow: Mir, p 270
[15-18] Mour E, Shennon K (1960) Reliable schemes of unreliable relays. In: Cybernetics collection. Moscow: Foreign litr., vyp. 1, p 109–149
[15-19] Trayon J (1966) Quadruplicate logic. In: Methods of redundancy introduction for computer systems. Moscow: Sov. Radio
[15-20] Elias P (1958) IBM Journal of research and development, No. 3, 1958, posterior probability, pp 346–353
[15-21] Winograd S, Cowan JD (1963) Reliable computation in the presence of noise, M.I.T. Press, Cambridge, Mass.
[15-22] Zrelova TI (1978) Review of methods for computer system reliability improvement. In: Computers and readjustable structure systems. Ser. Cybernetics problems, p 152–163
[15-23] Malev VA (1978) Structure redundancy in the logical devices. Moscow: Sviaz, p 192
[15-24] Bennets RG (1978) Designing reliable computer systems. The fault-tolerant approach-I. Electron and Power, posterior probability, pp 846–851
[15-25] Chernyshov Yu A, Abbakoumov IS (1979) Computing and design of computer devices with passive reservation. Moscow: Energiya, p 119
[15-26] Mathur FP, de Sousa PT (1975) Reliability modeling and analysis of general modular redundant systems. IEEE Trans. Reliability, vol. R-24, posterior probability, p 296–299
[15-27] Mathur FP, Avizienis A (1970) Reliability analysis and architecture of a highly redundant digital system. Generalized triple modular redundancy with self repair. Proc. SJCC, vol. 26, posterior probability, p 375–383
[15-28] Pakoulev NI, Ukhanov VM, Chernyshov PN (1974) Majority principle of design of units and devices of digital computers. Moscow: Sov. Radio, p. 183
[15-29] Losev VV (1971) Restoring organs on the basis of majority elements. Izv. AN SSSR, Technology cybernetics 2:116–122
[15-30] Fomin Yu I (1980) About restoring organs realizing majority voting. Electronic modeling 2:53–60
[15-31] Fomin Yu I (1980) Program for localization of parametrical failures. Annotated checklist of new receipts. MosFAP ASU 2:10
[15-32] Fomin Yu I (1980) Program for generation of minimum fault detection test. Ibidem, p 11
[15-33] Fomin Yu I (1980) Program for investigation of catastrophic reliability. Ibidem, p 11–12
[15-34] Fomin Yu I (1980) Program for investigation of parametrical reliability. Ibidem, p 12
[15-35] Fomin Yu I (1980) Program for the calculation of the failure probability of logical device consisting of duplex identical blocks with the network of majority elements at the input. Ibidem, p 13–14
[15-36] Fomin Yu I (1980) Program for the calculation of the failure probability of logical device consisting of duplex identical blocks with the network of majority elements at the output. Ibidem, p 13
[15-37] Potapov VI, Palianov IA (1973) Design of fault detection tests for the threshold elements. Izv AN SSSR, Technology cybernetics 4:140–146
[15-38] Fomin Yu I, Galushkin AI (1982) Majority voting and restoring organs for its implementation. Cybernetics and computer facilities. Vyp. 55, Kiev, Naukova Dumka, p 91–97
[15-39] Mkrtchan SO (1977) Design of computer logical devices on the basis of neuron elements. Moscow: Energiya, p 199
[15-40] Galushkin AI, Fomin Yu I (1979) About optimality of restoring organs realizing majority voting. Tekhnika sredstv sviazi, ser. ASU 3:56–61

Chapter 16

Neural Network Diagnostics

The introduction of dynamic redundancy into the structure of digital devices for reliability improvement requires the development of technical diagnostics of failures that occur in the structures implementing these devices.

Such diagnostic methods proposed in several works can be divided into two groups: methods of diagnostics and control of separate neurons at the level of separate functional components (multiplier, adder) and methods of diagnostics of neural networks at the level of separate neurons. The algorithms of the first group are described in detail in [15-15, 15-37, 16-1 to 16-4]. The authors made a complete classification of neuron failure types and developed synthesis algorithms for tests of neuron failure control and localization to an accuracy of the input-output. The algorithms of the neuron synthesis without logically indistinguishable failures to an accuracy of the input-output are developed in [15-15]. The proposed algorithms are sufficiently efficient for the separate neuron diagnostics, but they are practically inapplicable to the neural networks with a large number of neurons.

The algorithms of the second group are described in [16-5, 16-6]. They include diagnostic methods of neuron circuits of special types (2-neurons combination, cascade circuit, etc.). The disadvantage of the proposed diagnostic procedures is their low practical applicability.

The algorithms of the neural networks' technical diagnostics that provide the control over their performance and failure localization to an accuracy of a separate neuron are described below. The algorithm of neuron failure localization in the neural network is based on the investigation of the network state graph represented in Sect. 16.1. This technique provides the algorithm of minimum fault detection test for failures of the logical constant type. The method of adaptive diagnostics presented in Sect. 16.4 is based on the synthesis of an adaptive diagnostic network in the form of a neural network. It localizes any failure of the logical constant type that occurs in the neural network at the neuron input-output during one cycle of the neural network performance.

All the diagnostic algorithms considered in this chapter can be divided into two groups by methods of their realization: software or hardware.

The following algorithms belong to the first group: the algorithm of failure localization in the neural network (Sect. 16.2) and the algorithm of the minimum fault detection test design for the failures of the logical constant type. The former algorithm is based on the construction and investigation of the neural network state graph. The latter algorithm is based on the construction of the minimum neural network state graph corresponding to the minimum fault detection test for the logical constant failure type at the neuron outputs.

The second group includes the method of adaptive diagnostics of failures of the logical constant type at the neuron inputs-outputs. This method is based on the modeling of all possible failures of some given type, learning sample creation, and synthesis of the adaptive diagnostics network.

16.1 Neural Network State Graph – The Main Notions and Definitions

Different methods for the description of the neural network functioning exist: analytical, structural, geometrical, etc. Though each of these methods provides a complete description of a given neural network, it mainly reflects one particular characteristic of the network functioning. We introduce below the notion for the neural network state that describes the logics of its functioning.

Definition 1. Let us call the value a_{ij} that represents the outputs of all the neurons of the i-th layer and satisfies the condition

$$a_{ij} = \left\{ a_{ij}^1, a_{ij}^2, \ldots, a_{ij}^{H_1} \right\} \quad , \quad a_{ij}^l = \{0,1\}$$

as a j-th node of the i-th layer of the state graph. Here H_i is the number of neurons in the i-th layer; a_{ij}^l is the output value of the j-th neuron in the i-th layer.

Definition 2. The state graph branch is a directed segment linking two state graph nodes and designated as

$$a_{ij} \to a_{lk} \quad , \quad l = i+1$$

Definition 3. The state graph nodes of the zero level representing the input variable values are called the tops of the state graph.

Definition 4. The state graph nodes of the W-th level (W is the number of neural network layers) are called the roots of the state graph.

Definition 5. A path in the state graph is an arbitrary chain consisting of nodes linked by branches according to the functioning of the neural network possessing a top and a root.

Definition 6. The state graph represents a tree-like directed disconnected graph composed of paths and nodes located at corresponding levels.

Statement 1. Let us show that the state graph completely describes the neural network functioning for all values of the input variable. Consider all the nodes of the state graph related to the arbitrary path:

$$a_{0j_0} \to a_{1j_1} \to \ldots \to a_{Wj_1W}$$

Since the node a_{0j0} represents the value of the input variable, and the nodes a_{iji} ($i = 1, 2, \ldots, W$) represent the outputs of all the neurons and all the layers in the order of the number increase, then the network functioning, i.e., its total response to the given

input action a_{0j0}, is completely determined. Since the state graph is the aggregate of all the possible paths, then it determines the neural network response to all the input actions.

Definition 7. Neuron failure is considered as critical if the fault that results in the emergence of the error at this neuron output (when one or several input values occur at the input of this neuron) finally results in the error at the output of the whole neural network. Neuron failure is considered as uncritical if any error at its output does not result in the error at the output of the whole neural network.

Definition 8. A path in the state graph is considered as a faulty one if it corresponds to the neural network with a critical failure. I.e., the fault path possesses the root corresponding to the wrong value of the logical function realized by the neural network.

Definition 9. A path possessing a required root, i.e., a path corresponding to the neural network without failures, or with uncritical failures, or with critical failures that do not influence upon the given neural network input value, is considered as a correct one.

Definition 10. A complete state graph is a graph with 2^n tops, where n is the dimensionality of the neural network input.

Definition 11. A closed region formed by hyperplanes (realized by neurons) and hypercube faces is called a hypercube compartition. Each compartition has its number determined by the number of the neuron outputs, i.e., the number of compartitions is the state graph node.

16.2 Algorithm of Failure Localization in the Neural Networks

The essence of the proposed algorithm will be explained in some particular examples. Let us first consider the case of the single-fold failures and then generalize the obtained results for the case of multiple failures.

Let us take the three-layer neural network with three neurons in the first layer, two neurons in the second layer, and one neuron in the third layer. The location of hyperplanes realized by the first-, second-, and third-layer neurons is shown in Fig. 16.1a–c respectively, where x_i^j is the output value of the i-th neuron in the j-th layer (at $j = 0$, it is an input variable value). The criss-crosses indicate the values that provide 1 at the neural network output, and circles indicate the value providing 0 at the neural network output. The numeric characters at each hyperplane indicate the number of neurons in the layer.

The full state graph corresponding to the considered neural network is represented in Fig. 16.2. Let us consider, for example, a parametrical critical failure of the first neuron in the first layer. The hyperplane positions in this case are represented in Fig. 16.3a–c. As a result of the failure neuron weighting coefficient changes, the hypercube top 110 appeared in another compartition with the number 100 (it was initially in compartition 000). This results in the neural network output error: not all the criss-crosses and circles are in the different compartitions. The full state graph corresponding to this failure is shown in Fig. 16.4. The dashed line indicates the one-valued branches of the fault paths.

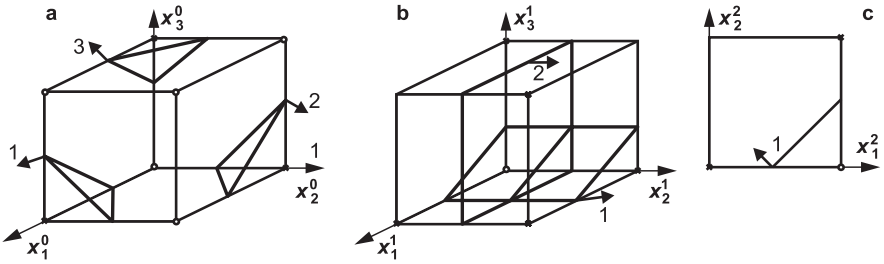
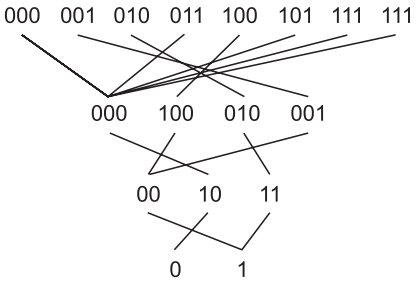


Fig. 16.1. Hyperplanes realized by the three-layer network without failures

Fig. 16.2.
The state graph of the free-failure three-layer neural network (Fig. 16.1)



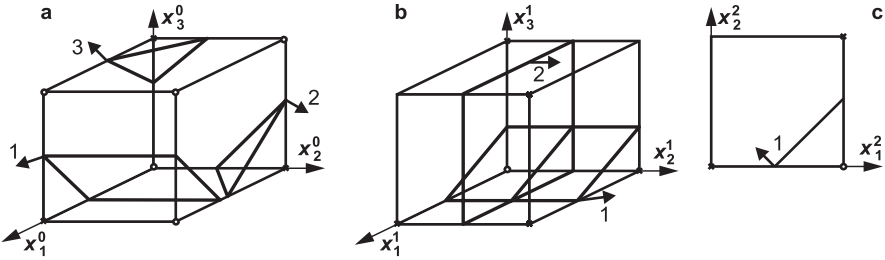
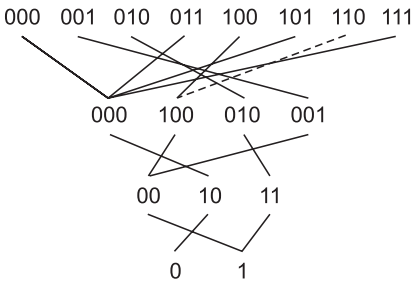


Fig. 16.3. Hyperplanes realized by the three-layer neural network with failure

Fig. 16.4.
The state graph of the three-layer neural network with failure (Fig. 16.3)



A single path in the state graph corresponds to each neural network input value. Consequently, the transfer of the hypercube top into the given compartition corresponds to the path transformation, related to this given top, into some fault path. This new path has the same top but some of its other tops are changed.

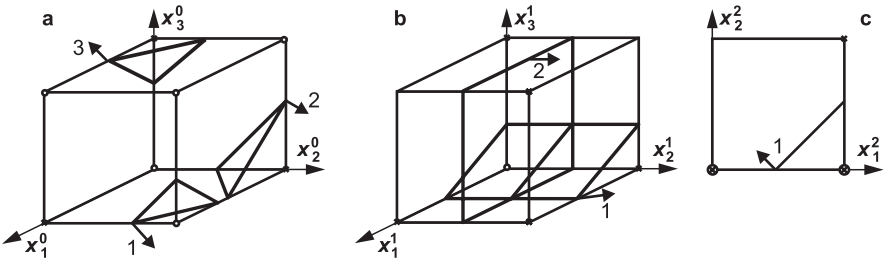


Fig. 16.5. Hyperplanes realized by the three-layer neural network with failure

Fig. 16.6. The state graph of the three-layer neural network with failure (Fig. 16.5)

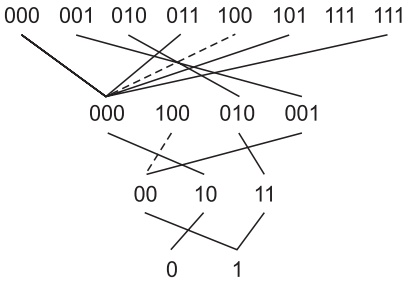


Figure 16.5a–c represents another example of the parametrical failure of the considered neuron. As a result of the fault, the tops 100 and 110 of the unit hypercube appeared in the new compartitions 000 and 100, respectively. It corresponds to the emergence of two fault paths in the state graph in Fig. 16.6.

The problem of the failure neuron search using the state graph consists in the search of such a transformation of one or several fault paths into correct paths that does not result in the emergence of additional fault paths. The form of this transformation (numbers of node positions in the state graph that changed their values) must indicate the number of the failure neuron. In the example represented in Fig. 16.4, the only fault path

$$110 \rightarrow 100 \rightarrow 00 \rightarrow 1 \tag{16.1}$$

must be transformed into another path

$$110 \rightarrow 000 \rightarrow 10 \rightarrow 0 \tag{16.2}$$

It is seen from (16.1) and (16.2) that some node positions changed their values. The first distinct position indicates the possible failure of the first neuron in the first layer. The validity of such a proposition can be shown if proving the following statement.

Statement 2. Only one-valued branches of the fault paths can be transformed with the help of the state graph at the search of the neuron failure in the neural network. Let us prove this. The values of the logical function (the roots of the state graph) are correctly and unambiguously put into correspondence with all the tops of the state graph corresponding to the failure-free neural network. Some of the branches change, and the

fault branch emerges at the failure occurrence. Each of the fault paths corresponds to a single error at the neural network output. The required transformation is the inverse to the failure emergence transformation, i.e., the transformation that eliminates the failure and does not add any new failure. Let us assume that we transform the many-valued branch of the fault path. This means that all the other paths possessing this node will obtain another root. And since these paths were correct according to the initial assumption, then the transformation converts them into the fault ones. But this contradicts the transformation feasibility, and therefore the statement is proved.

In the case of another failure in the state graph shown in Fig. 16.6, one observes two fault paths with the tops 100 and 110. The first path has the single one-value branch at the zero level. Let us consider all the possible transformations of this branch into the branches of the corresponding parts of correct paths: $(\rightarrow 100 \rightarrow 00 \rightarrow 1)$ and $(\rightarrow 010 \rightarrow 11 \rightarrow 1)$. As a result, the transformations have the following form:

$$100 \rightarrow (0 \rightarrow 1)00 \rightarrow 00 \rightarrow 1 \tag{16.3}$$

$$100 \rightarrow (0 \rightarrow 1)0 \rightarrow 11 \rightarrow 1 \tag{16.3}$$

where two values in brackets indicate the value change at this position during the transformation performance. One can expect from (16.3) that the failure neuron is the first neuron in the first layer. Similarly, one can expect from (16.4) that the failure neuron is the second neuron in the first layer. It can be written for the branch of the zero level in the second fault path:

$$110 \rightarrow (1 \rightarrow 0)00 \rightarrow 10 \rightarrow 0 \tag{16.5}$$

i.e., the first neuron of the first layer is under a cloud. The second path is also one-valued at the first layer. Consequently, the following transformation is possible:

$$110 \rightarrow 100 \rightarrow (0 \rightarrow 1)0 \rightarrow 0 \tag{16.6}$$

This suggests the possible failure of the first neuron in the second layer. Let us prove the following statement in order to reveal the required neuron out of the whole assemblage of the suspected neurons.

Statement 3. Let us assume that one critical neuron failure exists in the neural network, and more than one fault path exists in the state graph. If the set of neuron numbers suspected to be the numbers of the failure neurons is obtained, then there exists a neuron number that is found in this set a maximum number of times, and it represents the number of the failure neuron. Let us assume that there are N fault paths and one critical neuron failure in the state graph. It follows from the nature of any failure that there always exists a transformation inverse to the failure. Let us assume that this transformation corresponds to the neuron number that is not maximal out of all suspected neuron numbers. It means that not all the fault paths are transformed (as it follows from the search procedure for the suspected neuron). And the latter conclusion contradicts the above assumptions.

According to the above statements, the first neuron in the first layer in the aforementioned example is the failure neuron because its number is found twice among the suspected neurons.

The search process of the failure neuron number can be simplified on the basis of the following statement.

Statement 4. Let us perform a sequential (from the top to the root) comparison between the fault path in the state graph and the corresponding correct path. The first distinct position indicates the number of the failure neuron. Let us prove this statement. According to the above statement 2, in the case of several one-valued branches in the considered fault path, it is necessary to search the individual transformation for each branch. However, we want to prove that it is sufficient to find the only one transformation for one correct branch belonging to the uppermost level.

Let us consider one failure in the neural network. Compare sequentially the nodes of the fault path in the state graph. Let the node a_{ij} be distinct, and the branches of the j-th and $(j + 1)$-th level be one-valued. The noncoincidence of the node a_{ij} indicates the error at the output of the i-th neuron in the j-th layer. Since the nodes of the previous layers coincided, then namely the i-th neuron in the j-th layer is a failure neuron.

Only a single failure occurred in the neural network according to the assumption. Then all other noncoincidences of the nodes are caused by the failure of namely the i-th neuron in the j-th layer.

Let us list the sequence of the main stages of the algorithm for the failure neuron localization in the case of the single-fold failure. The state graph of the correct functioning neural network is considered to be given.

1. The values of the input variable are sequentially applied to the neural network input;
2. The neuron outputs are stored for each input value (a path in the state graph is created);
3. The obtained output value is compared with the root of the corresponding path in the given state graph (a path with the same top). If the roots coincide then go to p. 1, otherwise go to p. 4;
4. The positions of both paths in the state graph are compared from the top to the root;
5. The first distinct position indicates the number of the failure neuron, and the process terminates because the failures are single-fold.

The proposed algorithm can be easily generalized for the case of multi-fold failures. The example represented in Figs. 16.5 and 16.6 shows that the failure can influence the appearance of errors in the following layers, and the comparison between nodes can result in the wrong consideration of the failure after-effect as the failure itself. The failure search algorithm in the case of many-fold failures is

1. The values of the input variable are sequentially applied to the neural network input;
2. The neuron outputs are stored for each input value (a path in the state graph is created);

3. The obtained output value is compared with the root of the corresponding path in the given state graph (a path with the same top). If the roots coincide and the applied input value is not the last one then go to p. 1, otherwise go to p. 4;
4. Positions of both paths in the state graph are compared from the top to the root;
5. The first distinct position indicates the number of the failure neuron. If the applied input value is not the last one then go to p. 1, otherwise, go to p. 6;
6. Perform the failure correction and go to p. 6.

If the failure neurons are located in W layers, then the process is repeated W times.

In order to estimate the operation speed of the proposed method, let us compare it with the enumerative technique consisting in the test of each neuron in the neural network with single-fold failures. The state graph of the correct functioning neural network is considered to be given. Let us create the state graph corresponding to the neural network with failures.

A. Apply sequentially all the input variable values to the neural network input and obtain a state graph path and an output value for each input value. Compare the result with a correct neural network (without failures). If both outputs coincide, then apply the next input value, otherwise go to p. B. Thus, 2^{H0} elementary operations of bit-by-bit comparison are performed after the application of the full set of values to the neural network input.

B. A fault path is detected in the state graph. Compare it with the corresponding path of the correct neural network. The first distinct position indicates the number of the failure neuron. If only one neuron failure exists, then

$$\sum_{i=1}^{W} H_i$$

of elementary comparisons is performed after the application of the full set of the input values. Then the maximum comparison operation number in the case of the failure localization algorithm can be expressed in the following form:

$$N_{1\max} = 2^{H0} + \sum_{i=1}^{W} H_i \tag{16.7}$$

The value N_1 reaches its maximum (16.7) because the suspected failure neuron possesses the last number in the last layer and the last fault path.

Let us consider now the neural network test using the enumerative technique. If the complete test for one neuron has a length of 2^{Ni}, and all the neurons must be tested (N_i is the number of neurons inputs in the i-th layer; $N_i = H_{i-1}$), then the number of elementary comparisons in the case of enumeration is

$$N_{2\max} = \sum_{i=1}^{W} 2^{Hi-1} H_i \tag{16.8}$$

The value N_2 reaches its maximum in the sense that the test length for one neuron is estimated as 2^{Ni}.

Let us show the validity if the inequality

$$N_{1\max} < N_{2\max} \tag{16.9}$$

Taking into account (16.7) and (16.8), one obtains for the inequality (16.9)

$$2^{H_0} + \sum_{i=1}^{W} H_i < \sum_{i=1}^{W} 2^{H_i - 1} H_i$$

Removing the summation symbols in the latter expression, one gets

$$2^{H_0} + H_1 + \ldots + H_W < H_1 2^{H_0} + H_2 2^{H_1} + \ldots + H_W 2^{H_W - 1} \tag{16.10}$$

Taking into account the evident inequalities,

$$\left.\begin{array}{l} H_2 < H_2 2^{H_1} \\ \cdots \\ \cdots \\ H_W < H_W 2^{H_W - 1} \end{array}\right\} \quad \text{if} \quad H_i \geq 1, \;\; i = 2, \ldots, W$$

the inequality (16.10) takes the following form:

$$2^{H_0} + H_1 < H_1 2^{H_0} \tag{16.11}$$

The cases of practical interest are $H_0 \geq 3$ and $H_{1\max} = 2^{H_0}$.

The expression (16.11) in this case takes the following form:

$$2^{H_0 + 1} < 2^{2H_0} \tag{16.12}$$

Taking the logarithm of (16.12), one gets $H_0 > 1$, which is always valid. The inequality (16.9) is therefore proved.

Thus, the proposed failure localization algorithm is always faster than the algorithms based on the enumerative technique.

Let us perform the similar operation speed estimations in the case of multi-fold failures. Consider the case of m failures in k layers. Additionally, let the neuron failure localization be the worst one: $k = k_{\max} = W$. Then the inequality (16.9) has the following form:

$$W\left(2^{H_0} + \sum_{i=1}^{W} H_i\right) < \sum_{i=1}^{W} 2^{H_i - 1} H_i \tag{16.13}$$

Since $W \le H_1$, then the inequality (16.13) takes the form

$$W\sum_{i=1}^{W} H_i < \sum_{i=2}^{W} 2^{H_i-1} H_i \tag{16.14}$$

In the case $H_1 \ge H_2 > 1$, $W \le H_1$, one obtains

$$W(H_1 + H_2) < 2^{H_1} H_2$$

and the inequality (16.14) takes the following form:

$$W\sum_{i=3}^{W} H_i < \sum_{i=3}^{W} H_i 2^{H_i-1} \tag{16.15}$$

The inequality (16.15) is valid if $H_1 > H_2 > \ldots > H_W$ and $H_1 = \ldots = H_W$. Then the inequality (16.13) is also valid.

Let us estimate now (by the lower-bound estimation) the relative speed gain at the transfer from the enumerative technique to the failure localization algorithm in the case of multi-fold failures:

$$\frac{N^*_{1\max}}{N_{2\max}} = \frac{W\left(2^{H_0} + \sum_{i=1}^{W} H_i\right)}{\sum_{i=1}^{W} 2^{H_i-1} H_i} \tag{16.16}$$

where $N^*_{1\max}$ is the lower-bound estimation for the failure localization algorithm operation speed in the case of multi-fold failures. If $H_1 = H_2 = \ldots = H_W$, then the inequality (16.16) takes the following form:

$$\frac{N^*_{1\max}}{N_{2\max}} = \frac{1}{H} + \frac{W}{2^H} \tag{16.17}$$

If $H_1 > H_2 > \ldots > H_W$, then one can use some average value $\bar{H}$ instead of H in (16.17). For example, one can take

$$\bar{H} = \frac{H_1 + H_W}{2} \quad \text{or}$$

$$\bar{H} = \frac{H_1 + H_2 + \ldots + H_W}{W}$$

It is evident that one can ignore the second summand in the sum (16.17) at sufficiently high values of H. Hence, according to the lower-bound estimation, the relative speed gain at the transfer from the sequential enumerative testing to the proposed failure localization algorithm increases linearly with the increase of the number of neurons in the layers.

16.3 Algorithm of the Minimum Test Design for the Failures of the Logical Constant Type at the Neuron Outputs

The proposed algorithm for the minimum fault detection test design has a restricted application field because this test checks not all the faults of the constant type, but only the faults of the logical constant types at the neuron outputs. It can be used only in the cases of failure-free neuron inputs.

Let us consider the neural network input space representing a unit hypercube divided by hyperplanes into compartitions. The considered hyperplanes are realized by the first-layer neurons. Let all the compartitions except one include one hypercube corner, and the selected compartition includes n corners. Let the number of compartition be $a_{ij|j=1}$

$$a_{i1} = \left\{ a_{i1}^1, a_{i1}^2, \ldots, a_{i1}^{H1} \right\} \quad , \quad a_{i1}^l = \{0,1\}$$

Let us assume that the failure of some neuron results in the change of some value a_{i1}^l. Then the compartition number for all n corners lying inside it will change because only the failures of the logical constant types are considered. Consequently, this failure appeared to be displayed at all n input values. However, it is necessary that one failure must be displayed at not more than one input value. Then the procedure for the full test minimization (the test with 2^n input values) consists in the search of compartitions including more than one corner and eliminating any excessive corners in this compartition in order to provide only one corner inside it.

Since the hypercube corners are simultaneously the tops of the state graph, then the process of the full test minimization described above is similar to the full state graph minimization. Let us consider the case represented in Figs. 16.1 (neural network) and 16.2 (state graph). The belonging of several hypercube corners to one compartition is reflected in the state graph by the fact that all the paths with these corners have common branches beginning from the first level. Let us eliminate such corners. There are five of them in the considered example:

(000, 011, 101, 110, 111) (16.18)

Let us compose one top, for example 000, from the assemblage (16.18). Then the state graph looks like that represented in Fig. 16.7.

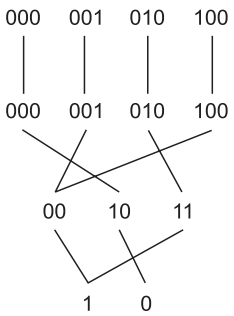


Fig. 16.7. Minimized state graph

The tops of this minimized state graph represent the input values for the minimized test

(000, 001, 010, 100) (16.19)

Let us prove that this test is minimal across the given above set of failures.

Statement 5. The number of tops of the neural network minimized state graph in the case of logical constant type failures is equal to the length of the minimum test disclosing all the failures of the given class.

The number of tops determines the length of the corresponding test according to the definition of the state graph. Let us prove now that the corresponding test is minimal for the given number of failures.

The state graph minimization consists in the sequential enumeration of the nonempty compartitions formed by the first layer neurons and in the elimination of the second, third, etc., input space hypercube corners belonging to one compartition. This procedure results in the state graph minimization corresponding to the state with only one argument value in each nonempty compartition.

Let us assume that the obtained test is not minimal. Then the elimination of any top from the obtained assemblage results in the emergence of a new empty compartition, and the logical function value in this compartition is undetermined. Therefore, the neural network minimized test cannot result in the emergence of the fault path corresponding to the eliminated top. Taking into account a single-valued correspondence between any path in the minimized state graph and some failure group, it appears that this failure group becomes undetectable at this test. This contradicts the test definition, i.e., the detectability of all the failures of a given class. Since the eliminated top was an arbitrary one, then the test corresponding to the minimized state graph is minimal, and the statement is therefore proved.

Taking into account all the aforesaid, the procedure for the minimal test design in the case of the logical constant type failures at the neuron output can be performed according to the following scheme:

1. Two levels of the full state graph (zero-order and first-order ones) are constructed for the correct functioning neural network. In this case, each input value corresponds to the first-level state graph node;
2. The number of non-recurrent graph nodes gives the length of the minimal test, and their corresponding tops provide the minimal test input values.

16.4 Method of the Neural Network Adaptive Failure Diagnostics

If the neural network must function in the continuous mode with high reliability, then it is impossible to interrupt its functioning for the diagnostic performance. The failure neuron therefore must be localized at the first application of the input signal that corresponds to the neuron failure. The method of the neural network adaptive failure diagnostics is used in this case. The adaptive diagnostic network is synthesized in the

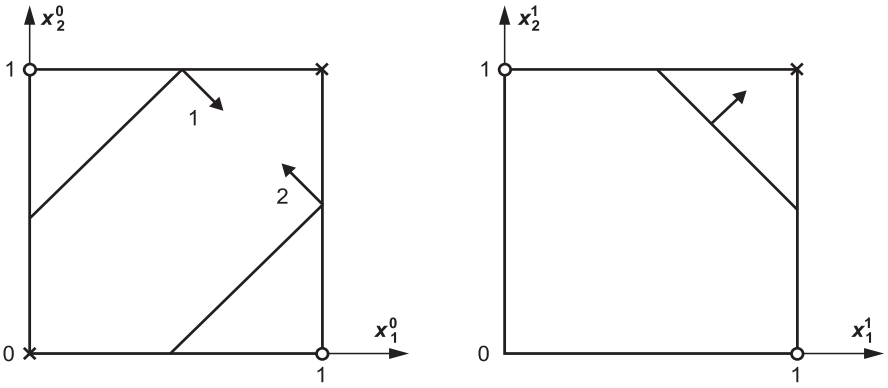
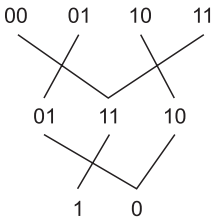


Fig. 16.8. Hyperplanes realized by the two-layer neural network

Fig. 16.9.
State graph of the two-layer neural network (Fig. 16.8)



form of the neural network with full sequential connections. It must be able to localize any failure of the logical constant type at the neuron input-output at the first application of the input value that corresponds to the neuron failure, i.e., to perform diagnostics parallel to the neural network functioning. This method can be therefore called as "a method of parallel diagnostics".

Let us consider the case of the learning sample generation for the adaptive diagnostic network synthesis in the example of the two-layer neural network with two neurons in the first layer and one neuron in the second layer (Fig. 16.8).

Let us assume here and below the existence of only one failure in the neural network. The test is carried out during one cycle of the neural network functioning. Figure 16.9 represents the full state graph of the considered neural network without failures. Figures 16.10–16.12 represent state graphs for all the possible failures of the given class, where x_{ij}^l is the value of the j-th input of the i-th neuron in the j-th layer, and x_{ki}^l is the value of the i-th output of the l-th layer.

Let us divide all the failures into classes corresponding to their neurons. The number of classes is equal to the number of neurons plus 1 (the last class is the class of the neural network without failures). Figures 16.10–16.12 represent state graphs of the two-layer neural network for all failures of constant types respectively of the first neuron in the first layer, of the second neuron in the first layer, and of the second-layer neuron output. Let us compose aggregates of the state graph fault paths (in the case of failure classes) or failure-free paths (in the case of failure-free classes). All the repeated paths in all the classes are excluded. Each fault path corresponds to one failure. All the fault paths are considered as a part of the learning sample that represents the failures classes.

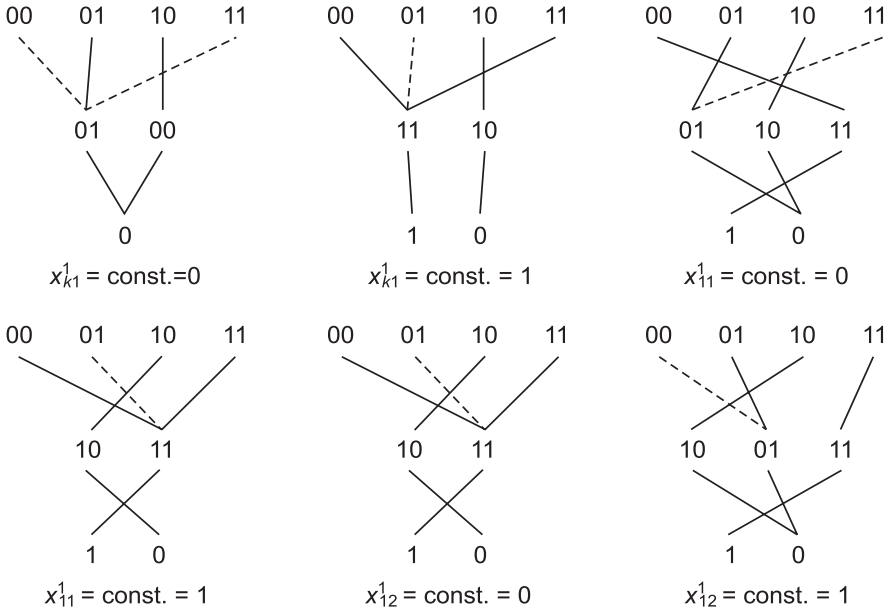


Fig. 16.10. State graphs of the two-layer neural network (Fig. 16.8) for all failures of constant types of the first neuron in the first layer

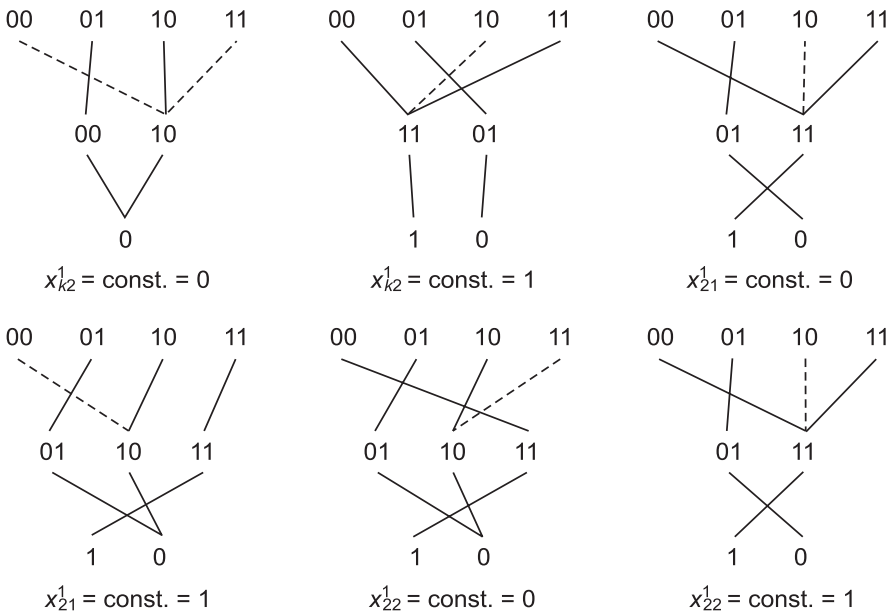


Fig. 16.11. State graphs of the two-layer neural network (Fig. 16.8) for all failures of constant types of the second neuron in the first layer

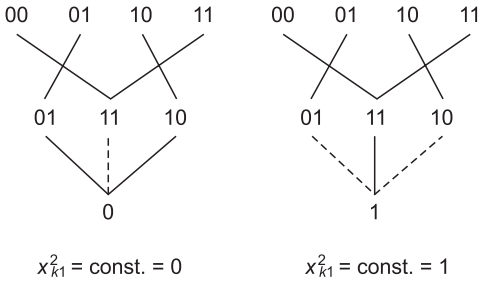


Fig. 16.12.
State graphs of the two-layer neural network (Fig. 16.8) for all failures of constant types of the second-layer neuron output

It has the following form in the considered example:

$$\left.\begin{matrix} 00\ 010 \\ 01\ 111 \\ 11\ 010 \end{matrix}\right\}1 \quad \left.\begin{matrix} 00\ 100 \\ 10\ 111 \\ 11\ 100 \end{matrix}\right\}2 \quad \left.\begin{matrix} 00\ 110 \\ 11\ 110 \\ 01\ 011 \\ 10\ 101 \end{matrix}\right\}3 \tag{16.20}$$

Let us take all the failure-free paths from the state graph represented in Fig. 16.9. They are considered as the second part of the learning sample that represents the failure-free neural networks class:

$$\left.\begin{matrix} 00\ 111 \\ 01\ 010 \\ 11\ 111 \\ 10\ 100 \end{matrix}\right\}4 \tag{16.21}$$

The numeric characters near the curly braces indicate the class numbers.

It is evident that all the paths in the state graph from the obtained aggregate (16.20), (16.21) represent the tops of the unit hypercube with dimensionality

$$N = \sum_{i=0}^{W} H_i$$

$N = 5$ in the considered example. The problem of the adaptive diagnostic network synthesis is solved as a usual pattern recognition problem. The neural network is synthesized using the learning samples (16.20), (16.21) by some adaptive algorithm. This neural network divides the unit N-dimensional hypercube into several compartitions consisting of the elements of one class. Such a partition is possible with the unit probability if there are no equal elements in different classes. Let us prove the following statement.

Statement 6. Any two paths in the neural network state graph are different if they correspond to the failures of two different neurons.

Let us consider two fault paths in the state graph:

$$\left\{a_{0j_0}, a_{1j_1}, ..., a_{Wj_W}\right\}, \left\{b_{0j_0}, b_{1j_1}, ..., b_{Wj_W}\right\} \tag{16.22}$$

They are equal under the following requirements:

$$a_{ij_i} = b_{ij_i} \quad , \quad i = 1,2,\ldots,W, \quad j_i = 1,\ldots,H_i$$

Consequently, the equal paths in the state graph have equal tops:

$$a_{0j_0} = b_{0j_0} \quad , \quad j_0 = 1,\ldots,H_0$$

Thus, in order to prove the statement, one must show that the corresponding failure-free path at the different failure neurons transforms into the different fault paths. Let this failure-free path be

$$\left\{c_{0j_0}, c_{1j_1}, \ldots, c_{Wj_W}\right\} \tag{16.23}$$

There are two different failure types: (1) the failures corresponding to the neurons of different layers and (2) the failures corresponding to the neurons of the same layer.

Let us consider the first case, when the failure neurons are located in the l-th and k-th layers. Then it is evident that $c_{ljl} \to a_{ljl}$, $c_{kjk} \to b_{kjk}$, where the arrows indicate to the transformation of one node to another one in the case of neuron failures. Thus, when the neuron failure in the failure-free path (16.23) occurs in the l-th layer, then the nodes c_{ij_i} with the numbers $l, l+1, \ldots, W$ transform into the nodes b_{ij_i} with the numbers $l, l+1, \ldots, W$ respectively. When the neuron failure in the failure-free path (16.23) occurs in the k-th layer, then the nodes c_{ij_i} transform into the nodes b_{ij_i} with the numbers $k, k+1, \ldots, W$ respectively. Consequently, one can write

$$\begin{aligned} c_{ij_i} &= a_{ij_i} \quad , \quad i = 1,\ldots,l-1 \\ c_{ij_i} &= b_{ij_i} \quad , \quad i = 1,\ldots,k-1 \end{aligned} \tag{16.24}$$

Let us assume that $l<k$, then according to (16.24), $a_{ij_i} \neq b_{ij_i}$, $i = l, l+1, \ldots, k-1, k$, i.e., the paths a and b are different.

Let us consider the second case, when the failure neurons are located in the same layer. Let the l-th and s-th neurons in the i-th layer be the failure neurons. Then

$$c^k_{ij_i} = a^k_{ij_i} \quad , \quad k = 1,\ldots,H_i, \quad k \neq l$$

$$c^k_{ij_i} = b^k_{ij_i} \quad , \quad k = 1,\ldots,H_i, \quad k \neq s$$

and consequently, $a^k_{iji} \neq b^k_{iji}$, $k = l, s$, i.e., the paths a and b are different.

Since there are no other neural network failure types beside the aforementioned ones, then the statement is proved.

As an example, the adaptive diagnostic neural network was synthesized using the learning samples (16.20), (16.21) according to an adaptive algorithm. Its block diagram is represented in Fig. 16.13.

The block diagram is reduced to the form of the neural network with full sequential connections. The connections with zero weights are not shown. The weights between

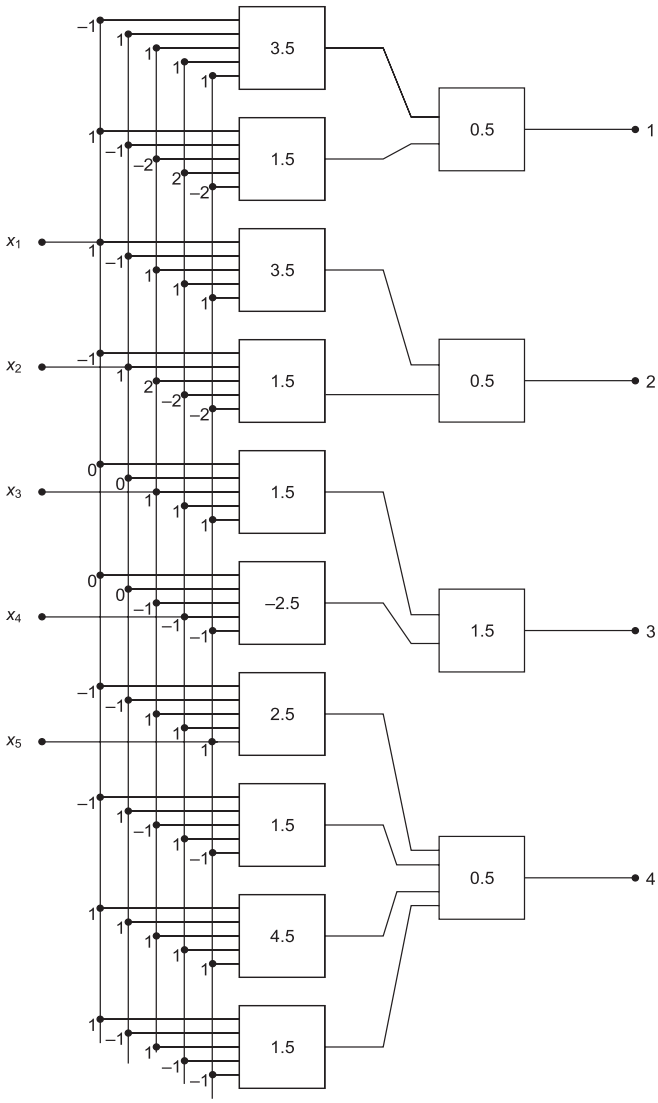


Fig. 16.13.
Block diagram of the adaptive diagnostic network in the case of the two-layer neural network (Fig. 16.8)

the neurons of the first and second layers are + 1. The weighting coefficients of the first-layer neurons are indicated near the corresponding inputs by the numerical characters. The threshold values are indicated inside the rectangles corresponding to the respective neurons. The neural network has four outputs, and it is synthesized in such a way that the emergence of 1 at one of the outputs (with 0 at all other outputs) means that the input value is considered to belong to the corresponding class.

The high structure redundancy observed at the adaptive diagnostic network synthesis is related to the disadvantages of this method. Such a redundancy is the "payment" for the high operation speed provided by such an approach. Another disadvan-

tage of this method is the existence of only one neuron at the neural network output. In this case, the failures at the outputs of neurons in the penultimate layer and the corresponding inputs of the last layer are logically undistinguishable. But the advantage of high operation speed makes this disadvantage insignificant for the neural networks with a large number of neurons.

Let us consider the peculiarities of the adaptive diagnostics method. The learning sample generation requires the modeling of all the failures of the given class. This is the most time-consuming part of the adaptive diagnostic neural network synthesis. The automation of this process allows one to simplify significantly the adaptive diagnostic network synthesis, and the use of the advanced algorithms allows one to obtain the optimal implementation of this network.

The realization of the neural network itself, as well as of the adaptive diagnostic network using mono-functional element with single-type connections, provides an advantage at the implementation of the whole device in the form of VLSI circuit. It also allows one to use one and the same synthesis technique for both networks. Since the neuron redundancy exists in the process of the adaptive diagnostic network synthesis, then it is advisable to use this method for production of devices with high reliability requirements.

The enhancement of the described approach application can be performed at the expense of the parametrical failure class. In order to prove this statement, one must show that all possible fault paths corresponding to the parametrical failures belong to the set of fault paths for the failures of the logical constant type at the neuron inputs-outputs. The check of this condition in the case of the simplest neural networks showed its validity. However, its proof in the general case seems to be rather complicated.

Literature

[16-1] Palianov IA (1975) Localization of failures in the threshold logical elements within the input-output precision. Radio-instrument engineering and microelectronics, vyp. 4, Omsk, p. 160–166

[16-2] Palianov IA, Potapov VI (1973) Failure diagnostics of the multi-functional threshold units. In: Mechanisation and automation devices for technical quality control, p. 73–83

[16-3] Palianov IA (1975) Design of fault detection tests for multi-threshold threshold elements. In: Computer facilities and control systems. Omsk, p. 70–78

[16-4] Potapov VI, Palianov I.A. (1976) Threshold element failure diagnostics. Izv. SO AN SSSR, ser. Tekhnicheskije nauki 8(2):126–133

[16-5] Charayev GG (1971) Technical diagnostics of threshold element circuits. Automatics and teleautomatics 1:151–158

[16-6] Charayev GG (1974) Technical diagnostics of discrete devices of integrated electronics. Moscow: Energiya, p 105

[16-7] Fomin Yu I, Galushkin AI (1980) Methods of technology diagnostics of threshold element networks. Tekhnika sredstv sviazi, "Sistemy sviazy" 2:84–94

[16-8] Fomin Yu I, Galushkin AI (1981) Methods of failure parallel diagnostics in threshold element networks. "Elektronnoye modelirovaniye", Kiev, Naukova Dumka, No. 3, p 89–92

[16-9] Sarje AK (1984) Implication chart for testing threshold functions. J. Inst. Eng. (India), Electron. And Telecommun. Engineering. Div." 65(2):46–77

[16-10] Mourad Samiha, Hughes Joseph LA, McCluskey Edward J (1986) Multiple fault detection in parity trees, COMPCON Spring 86, 31 step IEEE Comput.Soc.Interval.Conf., San Francisco, Calif., p 441–444

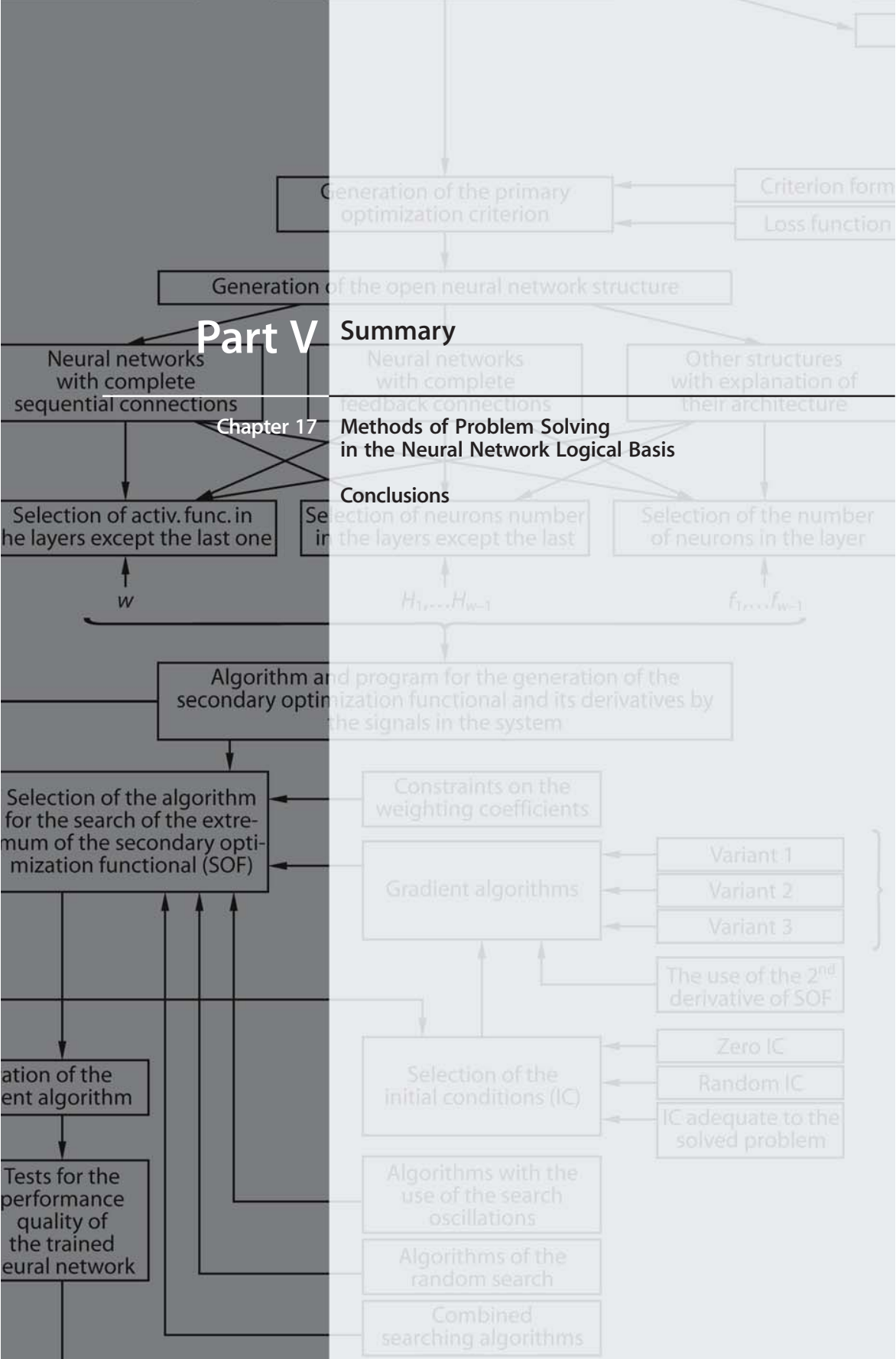
[16-11] Brown DP (1992) Matrix tests for period 1 and 2 limit cycles in discrete threshold networks, IEEE Trans Syst. Man and Cybernetics 22(3):532–534

[16-12] Eijkman Eg GJ (1992) Neural nets tested by psychophysical methods, Neural networks 5(1):153–162

Part V Summary

Chapter 17 Methods of Problem Solving in the Neural Network Logical Basis

Conclusions


Generation of the primary optimization criterion
Criterion form
Loss function
Generation of the open neural network structure
Neural networks with complete sequential connections
Neural networks with complete feedback connections
Other structures with explanation of their architecture
Selection of activ. func. in he layers except the last one
Selection of neurons number in the layers except the last
Selection of the number of neurons in the layer
w
$H_1,\ldots H_{w-1}$
$f_1,\ldots f_{w-1}$
Algorithm and program for the generation of the secondary optimization functional and its derivatives by the signals in the system
Selection of the algorithm for the search of the extremum of the secondary optimization functional (SOF)
Constraints on the weighting coefficients
Gradient algorithms
Variant 1
Variant 2
Variant 3
The use of the 2nd derivative of SOF
ation of the ent algorithm
Selection of the initial conditions (IC)
Zero IC
Random IC
IC adequate to the solved problem
Tests for the performance quality of the trained eural network
Algorithms with the use of the search oscillations
Algorithms of the random search
Combined searching algorithms

Chapter 17

Methods of Problem Solving in the Neural Network Logical Basis

17.1 Neuromathematics – a New Perspective Part of Computational Mathematics

Neuromathematics is a branch of computational mathematics dealing with the development of methods and algorithms for the solution in the neural network basis. The objective reason for providing the development of this new part of computational mathematics is a 30-year stock in the field of neural network theory that allows for the development of the universal approach of the neural network algorithms aimed at the solution in the domain of general and applied mathematics.

We shall call the computational procedure that can be realized mainly by the neural network of a different structure the *neuron algorithm* (or *neural network algorithm*). The main task that solves by the neurocomputer is the fast problem solving.

The first attempts to solve the computational problems with the help of neurocomputers relate to the 1960s and 1970s when the pattern recognition task was actual and included the problem of function approximation (*K*-classes of patterns in the multi-dimensional feature space). Thereafter, some other attempts to solve the classical computational problems with the help of neural networks were taken. One of the examples of such tasks is the matrix inversion problem. The number of problems solved with the help of neurocomputers was significantly enlarged at the end of the 1980s. One can tell now about the potential universalism of neurocomputers. It is clear that any mathematical problem can be solved on the neural network logical basis.

Even such problems that seem to be trivial (addition, multiplication, division, extracting a root, numerical inversion, etc.) can be solved with the help of neurocomputers much more effectively than with the help of the usual Boolean elements.

The field of application of the tasks that can be efficiently solved by neurocomputers is permanently and rapidly widening. The class of general mathematic tasks that can be efficiently solved by neurocomputers is rather wide. It includes, for example, the following kinds of tasks:

- Systems of linear and nonlinear algebraic equations and inequalities;
- The tasks of function approximation and extrapolation;
- The optimization tasks (linear, nonlinear, and dynamic programming; combinatorial tasks; the commercial traveler task; the task of the timetable arrangement; different tasks with graphs);
- The solution of ordinary nonlinear differential equations;
- The solution of differential equations in partial derivatives.

Various transformations can be realized in the neural network logical basis whereas their implementation using classical computers requires the development of special algorithms (algorithms of direct and inverse trigonometric and exponential functions that are the activation functions in the neural network approach).

An especially important part of general neuromathematics is a complex of problems related to graphs. In particular, these are the problems of the search formalization and calculation of the routes, cycles and cutsets in graphs, and the problems of partition of graph, its drawing and arrangement.

It is evident that the class of problems of general neuromathematics will increase in the nearest future. In the world, the number of scientific studies related to the neural network algorithms is rather large. The peculiarity of the Russian school of neuromathematics is the use of effective scientific results in the field of neural networks and corresponding effective methods for the neural network algorithm development adequate to the specific solution.

The development of neuromathematics was initiated not by mathematicians but by the specialists in the theory of control and neurocomputers under the "nonverbal behavior" of the single-functioned workers in the field of computational mathematics. The methods of control theory, analytical self-adaptive systems and adaptive filtering formed the basis of the development of the neural network algorithm methods. The neural network algorithms are rather "similar" for different mathematical tasks because a large number of common problems are present in the methods of neural network algorithm development in the wide variety of solutions. The majority of these common problems are either not taken into consideration or passed over in silence.

The objective reasons for the transition to the neural network algorithms for the solution of different tasks are the following:

- The inability to solve complex tasks of general and applied mathematics in a given time by the use of computational systems of other architectures (at the equal cost of neurocomputers and these computational systems);
- The objective necessity for the use of the neural network algorithm and its adequacy to the task under consideration.

This results in the following main distinctions between the neural network algorithm and any other one:

- The super-high parallelism (the parallelism of the neural network algorithms is always higher than that of technical facilities for their implementation);
- The high capacity-to-cost (or capacity-to-size) ratio of technical facilities for the neural network algorithms implementation.

As the development of neuromathematics (a part of computational mathematics realizing the tasks of general and applied mathematics in the neural network logical basis) is in progress, neurocomputers will pretend to the role of universal computational systems.

The main reasons to write the present article are the following:

- The primitiveness of the neural network algorithm that is used in the initial stages of the tasks' solution (as a rule, after some solution quality is achieved, there are no methods for its further improvement);
- The necessity for the elaboration of the neural network algorithms adequate to the task under consideration in the framework of some unified tasks solution technique in the neural network logical basis.

A large number of the known works in the domain of the neural network theory, neuromathematics, neural control and neurocomputers can be conditionally divided into two parts. *The first one* resulted from the general reflections of different authors who are interested in these problems and who want to improve the solutions which they have found in the scientific publications or which they invented on the basis of some general ideas. *The second one* deals with the development of ideas that are born in the process of specific problem solving. The long-term practice shows that the real and serious theoretical statements and studies in this field of activity are developed namely in the *second part* of these works. This fact is not a simple appeal for the more active solution of practical problems but is a result of the long-term analysis of the large number of theoretical studies in this field of knowledge.

The present study defines in some sense the logical pathway for the development of the neural network algorithms for the problem solutions and can serve as a basis for the creation of the intellectual program package implementing the neural network program solution algorithms.

17.2 Neural Network Theory – A Logical Basis for the Development of the Neural Network Problem Solution Algorithms

The neural network theory presents the logical basis for the solution of the tasks of general and applied mathematics in the same way as earlier Boolean logic was the basis for the solution of the tasks by the computers with Neumann architecture.

The neural network is a network with a finite number of layers consisting of single-type elements. Each element is similar to the neuron with different types of connections between layers. The number of neurons in the layers must provide the given quality of the solution, and the number of layers must be as small as possible in order to minimize the time for this solution.

The main properties of the neural networks are given below:

- The homogeneous neural networks are characterized by the gradual degradation due to the breakdown of separate elements. This fact was shown by Rosenblatt who constructed the three-layer perceptron with random connections in the first layer with a redundant number of elements in this layer. Here, the function realized by the neural network is distributed across the structure;
- The structure of the homogeneous neural network provides the possibility of large-scale parallelism during the performance of the large number of synchronous operations (addition, multiplication, and nonlinear fast-response transformation. The neural network structure does not contain complicated and "long" irrational opera-

tions over the operands (division, extraction of root, etc.) that the algorithms used in the monoprocessor computers;
- The neural networks implement a rather flexible and complex functional transformation of the input state space into the output one. Hence, the flexibility of this transformation can be controlled by the number of layers and type of the used connection.
- The neural network structure allows for the analytical description of the input space transformation into the output one;
- The previous property of the neural network structure allows for the analytical adjustment of the neural network and for the control of the algorithm functioning in the process of solution;
- The complexity of the neural network used for the solution of the particular problem reflects the complexity of the problem itself;
- In the future, with the use of the linear serial Gill machines, the neural network structure will allow for the solution of the problem of analytical description and the design of the adaptation algorithm synthesis in multilayer neural networks.

The main advantages of the neural networks being the logical basis of the complex problem solution algorithms are the following:

- The invariance of the neural network synthetic procedure, respectively the feature space dimensionality and size;
- Correspondence to the modern and cutting-edge technology in microelectronics;
- Fault-tolerance in the sense of monotonous, rather than catastrophic quality changes sums depending on the number of hors de combat elements in the sense of monotonous, rather than catastrophic problem-solution quality changes depending on the number of the breakdown elements.

The postulatory base of the neural network theory is the stochastic Bayesian model of the outward things. In this connection, the formation input signal is carried out in terms of the pattern channel and the channel of supervisor instructions. Additionally, the input signal represents in general the nonstationary random signal with a complex, unknown, multimodal density of probability distribution.

17.3 Selection of the Problems Adequate to the Neural Network Logical Basis

The bottom-line goal of the present study is the design of the program package in the neural network logical basis. The use of such a program package is necessary when the development engineer working over the solution algorithm has already finished the stage of decision concerning the necessity of the neural network approach and has assured himself that such an approach is necessary for him.

The investigator begins the development of the neural network algorithm from the physical problem statement. The description of the physical problem statement must be performed not verbally, but in the form of the specific document, describing the initial data and the essence of the physical result that must be obtained as the result of calculations. If the development engineer of the neural network algorithm gets the

physical problem statement from some other person, then he must update the problem statement together with the problem originator. At any rate, the problem originator must not necessarily be a specialist in neural network algorithms.

All the problems in the physical (not mathematical) problem statement are divided into two parts: unformalized and the formalized problems. Unformalized problems are the problems that cannot be formalized in the form of some mathematical terms, formulas, structures, graphs, etc. As it was mentioned above, the number of such problems permanently grows. These problems are usually complex and hypercomplex and they can be solved only in the framework of the neural network approach. The formalized problems are the problems that can be represented in the form of the system of linear or nonlinear algebraic equations, or in the form of the system of ordinary nonlinear differential equations, in terms of the system of equations in partial derivatives.

The question concerning the class of problems that can be solved in the most efficient manner by different computing devices designed according to the new principles is always topical. It was considered for a long time that neurocomputers are efficient in the solution of the unformalized or ill-formalized problems that obligatorily include the algorithms with the learning procedure using real experimental data.

The problem of approximation of the particular functions with the discrete domain of variation is one of the main problems of this type. This is the problem of pattern recognition. The unformalized problems are evidently an important argument for the use of neurocomputers. However, it is necessary to remember that the problem of pattern recognition is only a special case of function approximation. And not statistical (regression models) but rather flexible nonlinear (neural network) approximation methods are used in this case.

At the present time, a new class of problems with pronounced natural parallelism has emerged (signal processing, image processing). This class of problems does not require the learning procedure using the experimental data. However, it is well represented in the neural network logical basis.

It is also efficient to use the neural network algorithms for the problems with the input information space (input data) generated by the Monte Carlo method, rather then analytically.

The question about the efficiency and necessity of the representation and solution of the problem class in the neural network logical basis is very important.

The first problem class was the problem of pattern recognition. A lot of different algorithms and different architectures of computers were used for its solution in the 1960s. In the 1970s and 1980s, the neural network algorithms for its solution became dominant.

The second problem class with the dominant use of the neural network algorithms is the problem of function approximation and extrapolation. At present, the main problem consists in the methods of the neural network solution algorithm development in each particular case.

The third problem class in which the advantages of the neural network algorithms are practically proofed is the problem of the dynamic system control or neural control. Two main tasks concerning the dynamic object identification (analysis) and construction of the correcting filters in the control loop (synthesis) that can be solved in the neural network logical basis make the use of the approximation methods unnecessary for the nonlinear differential equation solutions that are oriented on von Neumann computers.

The aforementioned problems can be conditionally divided into two groups: the first one is adequate to the neural network logical basis and the second one is "general". The time required for the solution of the "general" problems of large dimensionality using von Neumann computers or transputer-like (claster) computers can exceed the admissible time. The achievement of the admissible time in this case can result in exceeding the capacity or the cost of the computer system. Then the necessity to develop the neural network algorithms and the neural network hardware emerges.

The necessity of the solution of mathematical problems of high dimensionality appears as a rule in the case of the solution of practical tasks related to high technologies in various scientific studies, industry and economics. Namely the widespread development and application of neurocomputers are an indication of the development of high technologies.

The problems that cannot be solved by the computational facilities of the current development level were always observed in the history of computer engineering. Generally, at the present time, the transfer to the neural network logical basis is used in the case of the sharp increase of the solution space dimensionality or in the case of the requirement of a sharp decrease of the solution time.

The iteration algorithms are the natural solution under the condition of the problem of high dimensionality. The known iteration solution algorithms in the neural network logical basis such as, for example, the algorithms for the solution of the systems of linear algebraic equations, are often rather primitive, consisting of only one layer. This decreases the problem solving quality. The use of the neural networks with different structures including the neural networks with feedback coupling opens a broad perspective of development in such a field of neuromathematics.

The increase of the class of problems solved in the neural network logical basis can be efficiently estimated by the ratio of the productivity rate to the cost as compared with the classical von Neumann computers. This estimation shows that neurocomputers are close to the class of general-purpose computers.

It is assumed that the algorithms and programs will be efficiently used on any existent and prospective neurocomputer and form the basis of the future mathematical program libraries for neurocomputers, i.e., the basis of the applied software for the prospective neurocomputers as general-purpose computers. The developed algorithm software will constitute the basis of neuromathematics.

It must be mentioned that the neural network solution algorithms for different problems are often "similar" to each other. They have a canonical neural network structure selected for some particular problem: the number of layers and the number of neurons in the layers, the neural network adjustment procedure. Therefore development engineers and users obtain the possibility for the objective quantitative comparison of the different algorithms.

We consider that any problem can be solved with the help of the neurocomputer much more effectively than with the usual computer due to the fact that any problem algorithm can be represented in the neural network logical basis with the controlled neural layer number and minimized number of iterations of the adjustment procedure.

This means that the neural network algorithm for the solution of any problem on the logical level is much more parallel than any of its physical implementation. This property differentiates neural computers from such systems as transputer-like ones in

which the software designers usually modify the solution algorithms developed initially for the single-processor computers. These modifications are aimed at minimizing the expenses related to the information exchange between processors in the problem solving process.

According to the aforementioned remarks, it is necessary to comment on Fig. 17.1 that represents the logical structure of the selection procedure for the problems adequate to the neural network logical basis. As it was mentioned, all the problems can be divided into two types as formalized and unformalized problems. The author considers that the unformalized problems can practically be solved only on the neural network logical basis.

After the development of the neural network solution algorithm for the unformalized problem, its programming on the workstation computer, and analysis of the solution time dependence on the neural network parameters (in particular, such a parameter is the problem dimensionality), one can determine if the time required for the solution is sufficient for the customer. If this time is sufficient enough, then the neurocomputer implementation is the workstation computer program. In the opposite case, one can choose, in practice, only two possible decisions:

1. The design of the hardware for the neural network problem solving accelerator based on some technology with dependence on the customer's requirements concerning the duration of the development work and concerning the weight, size and cost of the hardware unit.

   In this case, for the particular selected technology of the neurochip and neuro-plate implementation, one can approximately calculate the number of these neurochips and neuro-plates in the hardware accelerator. Then the neuro-plate, the unit or the pillar of the neural network hardware accelerator with the host-computer, represents the neurocomputer's implementation.
2. In the case of the strict requirements concerning the duration of the development work and the absence of the requirements concerning the weight, size and cost of the hardware unit, the development of the program for the claster computer with parallelizing of the neural network algorithm using several processors can be done. The number of processors required for the problem implementation in this case can be approximately estimated because the use of the neural network algorithms allows for the control performed by the neural network algorithm. The algorithm can provide smoothness of the processors' loading and minimize the expenses related to the information exchange between processors. Then the neurocomputer implementation is the program for the claster computer realizing the parallel neural network algorithm.

Computational mathematics deals with the solution of formalized problems. And if the customer is satisfied with the operation speed that provides the solution algorithm in the classical logical basis adequate to the von Neumann architecture then there is no necessity in the use of the neural network logical basis. The development of advanced technology and the complexity of the formalized problems due to the increase of the dimensionality often result in the unsatisfactory time required for the solution on the workstation computers with the use of the classical algorithms.

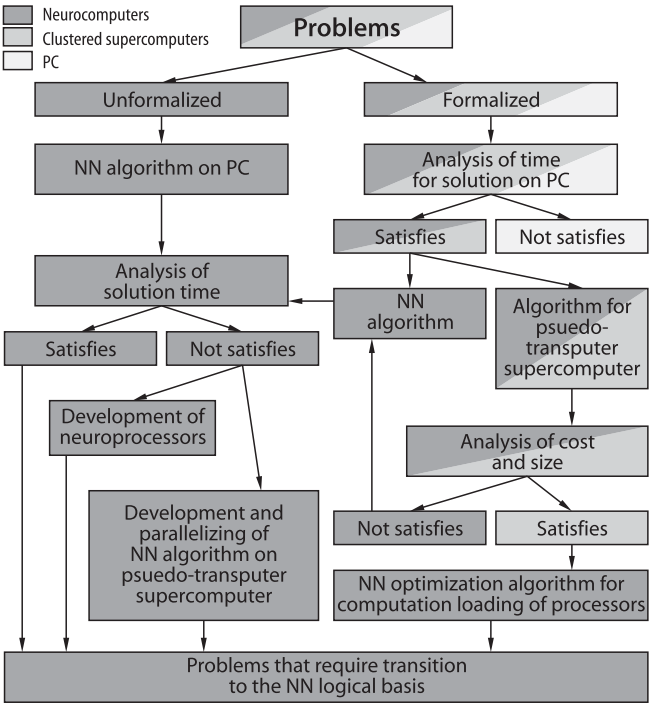


Fig. 17.1.
Logical structure of the selection procedure

Then, as it was mentioned above, the designer has only two possible decisions:

1. To develop the neural network algorithm for the formalized problem and then use the procedure described above for the unformalized problems, see [17-4], and references in that study, see also [17-6] and references in that study given in the section "Neuromathematics;"
2. To develop the program for the claster computer with parallelizing of the classic algorithm. This method is used by the majority of the claster computer users. However, one must take into account that the designer is solving
   - Either a purely scientific problem without constraints on the weight, size and cost of the computer, i.e. using the claster computer that is available to him;
   - Or the practical problem with significant constraints on the weight, size and cost of the computer. Then the designer is often forced to use the neural network logical basis and to develop the neurocomputer, i.e., to elaborate the neural network solution algorithm.

Notice that even in the case of the solution of purely scientific problems without constraints on the weight, size and cost of the computer, it is sometimes necessary to use the neural network logical basis for the solution of the problem of optimization of the loading distribution between processors of the claster computers. Therefore, Fig. 17.1 shows a set of possibilities for the neurocomputer implementation with the reasonable selection of the problems adequate to the neural network logical basis.

17.4 The General Structure of the Program Package for Problem Solution in the Neural Network Logical Basis

The foundation of the unified method for the problem solution in the neural network logical basis is the method of the adaptation algorithm synthesis for the multilayer neural networks. According to this method, the following adjustment algorithms for the multilayer neural networks were developed:

Neural networks for the general performance mode (learning, self-learning, learning with the supervisor of the finite qualification, etc.);

Neural networks for the wide class of the primary optimization criteria (minimum of the average risk function, average risk function under the constraints on its components, maximum of the a priory probability, maximum of the a-posteriori probability, etc.);

Neural networks for the wide class of the secondary optimization functionals (gradient, gradient with memory, combination of the gradient procedure with the random search for the initial condition selection, etc.);

Neural networks for the different multilayer neural network structures (with the arbitrary number of neuron layers, with complete sequential, cross and feedback connections, etc.).

The following principles form the basis of the neural network solution algorithm development:

- The refusal from the known neuro-packages of neural network programs and paradigms;
- The synthesis of the neural network algorithms adequate to each particular mathematical problem;
- The synthesis of the neural network algorithms and structures of the tuned neural networks without intrusion from the stated problem but with flexible and desired structure selection aimed at the improvement of the problem solving quality.

We consider the *problem solving quality* as the precision of the solution and the operation speed determined, in particular, by the number of iterations in the adaptation procedure of the neural network.

The general methods of the mathematical problem solution in the neural network logical basis were described in [17-2]. The neural network solution algorithms are represented in this study in the whole structure defined by the methods of synthesis of the multilayer neural networks that include the following stages of the problem statement:

- Physical, geometrical;
- Mathematical;
- Neural network.

The neural network problem statement includes in turn the following stages:

- Description of the initial data;
- Determination of the input signal $x(n)$ of the neural network;
- Generation of the primary optimization functional of the neural network for the solution of the problem;
- Determination of the output signal $y(n)$ of the neural network;
- Determination of the desired output signal of the neural network;
- Determination of the neural network error signal vector for the solution of the problem;
- Generation of the neural network secondary optimization functional through the signals in the system;
- Selection of the secondary optimization functional extremum search method;
- Analytical determination of the transformation performed by the neural network;
- Selection of the particular structure of the neural network;
- The search of the analytical expression for the gradient of the secondary optimization functional through the adjustment parameter;
- Generation of the neural network adjustment algorithm for the solution;
- Selection of the initial conditions for the neural network adjustment;
- Selection of the typical input signals for the verification of the solution procedure for the problem;
- Development of the plan of experiments.

The aforementioned stages of the neural network solution algorithm synthesis determine the complete circuit diagram of the user work with the program package (Fig. 17.2).

Figure 17.2 (see p. 372/373) represents the current version of the general structure of the program package for the solution in the neural network logical basis. This structure is the pathway for the development of the neural network solution algorithm and can serve as the basis for the design of the menu for the considered program package.

After the designer's decision to use namely the neural network solution algorithm, he can use the two following types of neural networks:

- Neural network with flexible (variable) structure [17-1];
- Neural network with the fixed structure.

The author believes that there is no third possibility at the present time.

It must be mentioned that in the Russian school of the neural network solution algorithms, the process of the solution for both possibilities was considered as some dynamical process with the use of some significantly nonlinear neural network environment. This approach was formed on the basis of the general theory of adaptive search and analytical systems.

17.5 Multilayer Neural Networks with Flexible Structure

The main advantage of the multilayer neural networks with flexible structure is the absence of the obligatory a priori information about the neural network structure (the

number of layers and the number of neurons in the layers). This structure is formed in the process of the neural network adjustment, i.e., in the process of the solution. The obtained structure indirectly reflects the problem complexity. The more complex the trained neural network is (the number of layers and the number of neurons in the layers), the more complex the solution is.

The neural networks with flexible structure [17-1], [17-2], and [17-3] can be efficiently used for different types of the input feature space:

- Binary, when the variables of the input N-dimensional vectors are represented by the set of zeros and units;
- K-digital variables of the N-dimensional vector;
- Real-valued variables of the N-dimensional vector.

The multilayer neural networks with flexible structure were mostly used for the case of the N-dimensional space of the real-valued feature when the input information represents the continuum signals from some fixed time interval.

The limitation of the multilayer neural networks with flexible structure is the fact that they can be used only for the solution of two problem classes:

- Recognition of two classes of patterns;
- Recognition of K classes of patterns (with generation of κK neural networks, recognizing each κ-th ($\kappa = 1, \ldots, K$) class from the other one);
- Self-learning (clasterization), when the input sample presented for the clasterization without the supervisor instruction about belonging to the different classes is the sample of the first class for the multilayer neural network with flexible structure, whereas the output of the white noise generator is the sample of the second class.

It is evident that the number of problems solved with the help of the multilayer neural networks with flexible structure will increase in the future.

In the procedure of the adjustment of the multilayer neural network with flexible structure, the first layer is trained at the beginning of the procedure when the required number of neurons H_1 of this layer is determined. Thereafter, the results of the first layer adjustment are used for the adjustment of the second and third layers. The number of layers in the solution of the two-pattern recognition equals 2 or 3 and only the single neuron is at the output.

Then the stages of development of tests for verification of the trained neural network quality are executed. Thereafter, the plan of experiments is elaborated for the investigation of the quality of the neural network performance. These stages are common for the multilayer neural networks with flexible structure, and they will be considered below after the consideration of the neural networks with fixed structure.

At the end of the section dedicated to the multilayer neural networks with flexible structure, one must notice that at the present time, these neural networks are used for the solution of the problems of a relatively narrow class including the recognition of patterns of two or K classes and clasterization (self-learning). There is a potential probability of using such neural networks for the solution of wider classes of the problems (function approximation and extrapolation, etc.).

17.6 Neural Network with Fixed Structure

According to author's opinion, in contrast to the neural networks with flexible structure, the neural networks with the fixed structure can be used for the solution of **any** problems in the case that they satisfy the aforementioned selection criterion No. 3. The restriction on the neural network structure selected a priori is the payment for this universalism in the problems that can be solved. The neural network structure in this case is one of the components of the vector that includes all the types of the a priori information that are required for the neural network solution algorithm development. The complete description of this vector will be given in the conclusion of the present study. The stages of the neural network solution algorithm development with the help of the neural networks with fixed structure are given below. These stages are the following:

1. Generation of the input signal including the formation of the supervisor instructions;
2. Generation of the output signal;
3. Formation of the primary optimization functional;
4. Generation of the open neural network structure;
5. Formation of the secondary optimization functional;
6. Formation of the search algorithm for the secondary optimization functional extremum;
7. Formation of the algorithm for the adaptation of the coefficients of the multilayer neural network with fixed structure;
8. Development of tests for the verification of the performance quality of the trained neural network;
9. Elaboration of the plan of experiments for the verification of the performance quality of the trained neural network.

The last two items in this list are common for the neural networks of flexible as well as fixed structures.

17.6.1 Generation of the Input Signal of the Neural Network

This problem is not trivial and sometimes it is not single-valued but has several solutions. It can be relatively simply formulated in the pattern recognition tasks where the patterns are already represented by the vectors of features. However, in the particular problems of signals or pattern recognition, the generation of these patterns is a rather complicated problem. This problem is complicated even in such a transparent task as the function extrapolation because of the introduction of the additional parameter (filter memory) and the special method for the further generation of the supervisor instructions for the neural network. Similar problems exist in the tasks of the neural network equalizer development, systems of neuron control, etc.

The input signal of the neural network is the signal $[\bar{x}(n), \bar{\varepsilon}(n)]$, where $\bar{x}(n)$ is the series of the input patterns, $\bar{\varepsilon}(n)$ is the supervisor instruction about the patterns $\bar{x}(n)$ belonging to a particular class. Thus, both $\bar{x}(n)$ as well as $\bar{\varepsilon}(n)$ can be represented in different ways with dependence on the particular problem statement. The series $\bar{x}(n)$

can be the vector of the real-valued variables, the function of some argument, some vector-function, etc.; the series $\bar{\varepsilon}(n)$ can be the real-valued variable that takes two, K or a continuum of values, or it can be the vector or some vector function.

The probabilistic approach to the perception of the outside makes it necessary to generate the input signal for the neural network in the form of the joint distribution function $f(\bar{x}, \bar{\varepsilon})$. The detailed form of this function in the different particular cases can be rather different. In the majority of the investigations of the neural network learning algorithms, it is assumed that the supervisor qualification is complete, i.e., the supervisor can exactly determine the belonging of the particular pattern to a given class.

17.6.1.1
About the Supervisor Qualification

However, in practice, the problems with the limited supervisor qualification also exist and they are not sufficiently investigated.

The elaboration of the neural network algorithms adequate to the real conditions of getting information for their adjustment requires an estimation of real supervisor qualification. Due to this requirement, along with the widespread learning modes of the multilayer neural network in which it is assumed that the supervisor is aware about the patterns' belonging to a particular class with unit probability, one must consider in more detail three more learning modes introduced in [17-2]:

- Learning with the supervisor having zero qualification (self-learning, clasterization);
- Learning with the supervisor having finite qualification;
- Learning with the supervisor having negative qualification (the "harm" mode in which the supervisor wittingly gives false information about the pattern belonging to a particular class).

Neural networks in the self-learning mode. Clasterization. In spite of the long-term history of this problem, it remains still poorly investigated. The main task here consists in the processing of the set of multi-dimensional vectors aimed at the selection, according to a certain rule, of compact vectors' groups termed clasters. The investigations in this domain were not activated during last decades due to the absence of the socio-significant problems in which the self-learning mode would play an important role. But today such problems have begun to appear. From our viewpoint, the most significant one is the problem of information compression (compression of images, speech information, etc.) the solution of which by the existent classical methods have achieved the real limits of its capacity. In the scope of this problem, the aforementioned tasks that accompany the problem must also be further developed. Namely these tasks are the following:

- The typical input signals;
- The initial adjustment conditions;
- The control over the iteration procedure parameters, etc.

Neural networks in the mode of learning with supervisor having finite qualification. Since the 1970s, when designing some specific systems of electrocardiogram recogni-

tion, it has been noted that in the process of electrocardiogram archive verification, an expert physician or a group of expert physicians cannot relate some electrocardiograms to a particular class of diseases with full reliability. The investigation of the dynamics and of the results of the multilayer neural network adjustment as a function of the real idea about supervisor qualification is an important subject for future investigations in the field of neural network theory.

Neural networks in the "harm" mode of learning. The neural network operation mode in which the supervisor wittingly gives false information about the pattern belonging to a certain class is completely unstudied. Probably this mode will be used under war information conditions in order to estimate the information safety of the corporate systems and regions with the help of multilayer neural networks in the case of information about weapon utilization.

Notice that in the known scientific literature, two different operating modes of the neural networks, namely the learning mode (the supervisor qualification is complete) and the self-learning mode (the supervisor qualification is zero), are regarded independently from each other. In the proposed methods, these two modes differ only by the value of some parameter. The variation of this parameter allows for the consideration of a lot of new modes.

17.6.1.2
Taking into Account A Priori Probabilities of the Classes' Emergence

The necessity of taking into account the a priori probabilities of the classes' emergence appears in different practical tasks. A typical example is the task of letter recognition in the printed text of the scanned document in the case when the probability of any letter appearance is known. The possibility to use the a priori probabilities of the classes' emergence during the adjustment of the multilayer systems for image recognition is investigated in the study [17-2]. This possibility was permanently used in that study for the construction of the different particular systems. However, this technique requires additional investigation in order to use it efficiently.

17.6.1.3
Continuum of Classes

The general representation of the input signal for the multilayer neural networks of two, K and continuum pattern classes in the learning mode with the limited supervisor qualification allows for the use of the neural networks in the problem of quantitative estimation of the object state described by the signal $\bar{x}(n)$. In this case, the series $\bar{\varepsilon}(n)$ is real-valued and varies in some limited interval.

17.6.1.4
About the Nonstationarity of the Input Signal

In the majority of the practical problems, the neural network input signal is considered to be stationary with some unknown and complex distribution function $f(\bar{x}, \bar{\varepsilon})$.

However, problems with the non-stationary input signal $f(\bar{x}, \bar{\varepsilon})$ exist, and it must influence the characteristics of the multilayer neural network adjustment algorithms. In the learning mode, the pattern distribution functions of each class are time dependent. In the self-learning mode (clasterization), both coordinates of the classes' centers as well as their characteristics can be time dependent.

17.6.2 The Multilayer Neural Network Output Signal Generation

The output signal of the neural network is formed according to the type of solved problem. This signal can be the binary value (or the vector of the binary values), K-digit value (or the vector of K-digit values) and real-valued variable (or the vector of the real-valued variables). The number of neurons in the output layer of the neural network and the form of the activation function of the neurons in the output layer are determined according to the output signal type. In the particular case, the output signal represents some spatial argument. The output layer in this case represents the neuron continuum with the real-valued output signals rather than the discrete set.

17.6.3 Formation of the Primary Optimization Criteria

The basis of the multilayer neural network primary optimization criteria includes the following items:

- The assumed probability concept of the external world;
- The consideration of the external world as a significantly nonlinear one.

Namely this basis allows for the formation of the primary optimization criterion as the main goal that the designer wants to achieve in the development of the multilayer neural network with the adaptation algorithm for the particular solution. The probability criteria described below are valid for the relatively wide class of problems, but this class can be further enlarged.

The recently developed methods for the multilayer neural network adaptation algorithm synthesis can be used for the following primary optimization criteria:

- $\min R = p_1 r_1 + p_2 r_2$ of the average risk function;
- $\min p_1 r_1$ under $p_2 r_2 = \text{const.}$;
- $\min R$ for K and continuum of classes;
- The aforementioned variants for two K_p and continuum of classes;
- Different modifications of the aforementioned criteria, for example, the criterion of maximum of the a-posteriori probability.

All these criteria can be used for the solution of the particular practical problems. One must consider the aforementioned criteria for the neural network synthesis in the procedure of the formation of the error-cost matrix. This matrix is used to make the decision that the pattern of one class belongs to another class (the error function in the continuum case).

As a rule, in the known studies, the error-cost matrices that emerge in the course of the assignment of the particular pattern to a particular class are assumed to be diagonal. However, it is not often in agreement with reality. For example, in the case of the neurocomputer design for the mine recognition system with the use of the geolocator, in the matrix of costs for errors

$$L = \begin{bmatrix} l_{11}, l_{12} \\ l_{21}, l_{22} \end{bmatrix}$$

the coefficients l_{21} (the cost for the error to consider the mine as an irrelevant object) and l_{21} (the cost for the error to consider the irrelevant object as a mine) cannot be equal in principle, and it must be taken into account during the adjustment of the multilayer neural network in a similar way to how it was carried out in the studies [17-2, 17-3].

The often used criterion of the minimum of the mean-root square error is the primitive and particular case of the aforementioned criteria.

17.6.4
Selection of the Open Neural Network Structure.

The a priori information about the neural network structure that is used in this stage of the neural network synthesis is the payment for the solution universalism. It is necessary to mention two main classes of the neural network structure that are used at the present time for the solution:

- Neural networks with complete sequential connections;
- Neural networks with complete feedback connections.

The selection of the neural network structure results in the following:

- The selection of the number of neuron layers;
- The selection of the number of neurons in all the layers except the last one (the number of neurons in the last layer is selected in the stage of the generation of the neural network output signal).
- The selection of the activation function in all the neuron layers except the last one (the activation function for the last layer is also selected in the stage of the generation of the neural network output signal).

17.6.5
Remarks about the Selection of the Open Neural Network Structure that is Adequate to the Class of Solution Tasks

In the majority of the scientific literature, the structure of the open neural network is introduced by the authors without any explanation. The main idea of the Russian works in this field is the development of the multilayer neural network adjustment algorithms adequate to the particular solution.

If the class of solution tasks allows one to define the class of effective neural network structures adequate to these tasks, then the elaboration of the special methods for the coefficient adjustment namely for this class of the neural network structures will increase the adaptation effectiveness for the task solution of this specific class. We describe below some variants of the neural network structures and the task classes adequate to them.

Neural networks with random connections. In his classical monograph, Rosenblat suggested introducing the random connections between the retina and the first layer of the multilayer neural network. Under the proper increase in the number of neurons, the systems' reliability, related to the possible break-down of several neurons, increases. At present and in perspective of the development of microelectronic technology, the number of emulated neurons inside the super-large-scale integration on the board and in the unit is quite large. And this number will continue to increase in the future. This makes the variant of the random connections more and more necessary for implementation and research.

Neural networks with lateral connections. This specific type of connections between layers in the multilayer neural network is interesting from the viewpoint of implementation of the invariance to the transformation group and has been poorly investigated. This is not only the invariance to the simplest affinity transformations, such as rotation, transition and the change of the affinity ratio, but also the invariance to the more complex transformations and the search of the connections' structures ensuring such invariance.

Cell-like neural networks. Cell-like neural networks are networks with a special topological structure that is adequate, in particular, to the task of pattern processing. In this case, the natural task parallelism results in the natural parallelism in the structure organization of the processing neural network. It is necessary to note that the cell-like neural networks are adequate to the other tasks with natural parallelism; for example, to the task of lattice generation and other tasks emerging at the solution of two-dimensional differential equations in partial derivatives. When changing from 2D- to 3D-tasks, the similar three-dimensional cell-like neural networks will be adequate to the tasks of three-dimensional pattern processing, 3D-lattice generation, virtual reality, and the solution of three-dimensional differential equations in partial derivatives.

Neural networks with feedback loops. The conception of the neural network with feedback loops in its classical sense was introduced in [17-2]. In this case, the feedback channels are present in the structure of the adjusted multilayer neural network. These channels are used for the transmission of the intermediate and output signals of the neural network to the input channels of the previous layers through the delay lines (for a given number of cycles). In the past, in the 1960s–1970s, it was considered that such neural network structures could be used only for the design of the memory units of a different functional destination. In the last ten to fifteen years, in foreign literature, the investigations of such neural networks, conventionally called recurrent ones, were sharply activated. Moreover, the range of their application increased including

the tasks of function approximation and extrapolation and the systems' dynamic control tasks. The neural networks with feedback (recurrent neural networks) are the natural control devices and nonlinear controlled object identification devices in the nonlinear systems of control. This is similar to the case of the linear systems of control in which *Z*-filter is the linear control device and *Z*-transformation is the formal description of the controlled object. The neural network with feedback channels is a typical example of how the structure of the adjusted multilayer neural network is selected from the criterion of adequacy to the solution task but not from the simple fact that an author is familiar with this or that neural network structure. In case of a neural network with feedback channels, such additional problems for investigations emerge as the determination of the degree of an equivalent decrease in the number of neurons in the adjusted multilayer neural network after introducing the feedback into its structure.

Neural networks with variable (flexible) structure. Since the 1960s, the multilayer neural networks with variable structure have been an effective tool in solving the task of pattern recognition [17-1]. This is the variant of the adjustment algorithms whose neural network structure (the number of neurons in the layers and the number of layers) grows in the process of adjustment up to a certain value of the solution quality index. The synthesis of the adjustment algorithms for the multilayer neural network with variable structure is a promising technique to solve a wide range of practical tasks.

Continual neural networks. Continual neural networks [17-3] are used mainly in two cases:

- When the number of indications in the layer is large;
- When the signal or pattern processing is performed on a real time basis and without preliminary quantification of the input information.

It is shown in [17-2] that the neural network adjustment algorithms under the continuum of indications or under the continuum of the neurons in the layer are the objects of independent consideration and research.

Complex neural networks. The input signals and the weighting coefficients in the neural networks of this type are represented in the form of complex numbers, and all the operations in the open neural networks and in the adjustment algorithms include the complex numbers. This type of neural network is widely used for nonlinear signal processing.

Interval neural networks. In this case, the input signals are determined not by their values but rather by the interval to which they belong.

17.6.6 Remarks about the Activation Function Selection

The activation function selection is an important element of the neural network synthesis procedure. More than ten types of the neuron activation function are described

in the known literature. Usually their selection is performed arbitrarily. In the studies [17-1, 17-2] the activation function $(2/\pi)\text{arctg}\, Bg$ was used, where g – is the analogous output signal of the neuron. At the end of the 1980s to the beginning of the 1990s, the sigmoid activation function became widely used. Wavelet and RBF networks are in fact the neural network with the activation function of a particular type.

As a rule, the introduction of the new peculiar type of the activation function represents an attempt to make the neural network be more adequate to the solution task in order to decrease the number of neurons and adjusted coefficients. However, the desired goal is not always achieved because the task of the neural network adjustment simplification is also desired, in addition to the task of the simplification for the calculations of the output signal from the input signal. It is necessary to note that the complication of the activation function results in the sufficiently sharp complication of the adjustment algorithm due to the fact that the computational units for the activation function derivatives' calculations are used in the adjustment algorithm.

On the whole, at present, the problem of selection of the neurons' activation function in the multilayer neural network is far from its solution.

17.6.7 Selection of the Multilayer Neural Network Structure According to its Hardware Implementation Technology

Some types of the open neural network structures are used due to the constraints of the neural network hardware implementation technology. The following neural networks belong to this class:

- The neural networks with cross-connections (from the i-th to the $i + 2$,… layers) for the decrease of the number of the realized neurons with some increase of the number of weight coefficients [17-2, 17-3];
- The neural networks that realize the feature continuum, the continuum of the neuron number in the layer, etc., for the implementation of the analog-to-digital neurocomputers and the signals and pattern processing [17-3];
- The neural networks with the weight coefficients of the finite digit capacity or with the adaptation algorithms for the weight coefficient digit capacity control.

The problem of the weight coefficient digit capacity for processing signals and patterns of different digit capacity representation is independent and very important. The low digit capacity of the weight coefficients results in additional errors, and the high digit capacity results in the high cost of the system and long processing time. This problem was correctly solved in the implementation of the linear z-filter hardware using very large-scale IC IMS A100 of the Inmos firm in 1986. These very large-scale IC z-filters have the digit capacity that can be programmatically changed in the range 2 … 16 with the corresponding decrease of the processing speed. In nonlinear filtering, this digit capacity can be changed according to some complex criterion. The problem of the adaptation algorithm development for the digit capacity control using very large-scale IC is the problem of future investigation. It emerges in the following domains:

- The neural networks with the Boolean values of the weight coefficients for the simplification of the implementation of the open neural network. This simplification increases the number of neurons in the solution of the given quality;
- The neural networks without multiplier units because the multiplier units represent the main difficulty of the hardware implementation;
- The neural networks with the limitations on the weight coefficients that must be considered during the formal description of the open neural network structure and during the adjustment algorithm development [17-2, 17-3].

The introduction of constraints on the open neural network structure is a very significant problem in the analogous, analog-to-digital, optical, molecular and quantum neurocomputers, as well as in the development of neurocomputers based on the single-electron nanocircuits.

17.6.8 Generation of the Secondary Optimization Functional in the Multilayer Neural Networks

In general, the primary optimization functional describes the neural network optimization criterion implemented in the hardware level. In contrast, the secondary optimization functional must be defined through the input and intermediate signals in the multilayer neural network and through the formal description of the open neural network structure. In the simple case, the goal of the secondary optimization functional development is the development of the analytical transformation at the neural network output that provides the second distribution moment of the signal corresponding or equal to the primary optimization functional [17-2, 17-3].

17.6.9 Generation of the Algorithm of the Search Procedure for the Secondary Optimization Functional Extremum

During the period of the 1960s and the beginning of the 1970s, neural networks were considered as the particular case of the nonlinear multi-dimensional object with the adjustable parameters implementing the adaptation self-learning control system.

In that period, the adaptation self-learning systems were developed mainly in the following two domains:

- *The search systems* with artificial search oscillations of the adjustment parameters for the following calculation of the optimization functional gradient;
- *The analytical systems* without artificial search oscillations of the adjustment parameters. The optimization functional gradient was calculated directly through the current input and output neural network signals as in the particular object of control.

The analytical methods of self-learning were developed mainly in Russian investigations, though in the theory and practice of the self-learning automated control, the search methods were dominant.

The following four variants of the secondary optimization functional gradient search were used before 1974:

- Gradient method;
- Gradient method with the time averaging of the gradient value estimation;
- Gradient method with the constraints on the neural network weight coefficients;
- Combination of the random selection of the search initial conditions and the gradient procedure for the search and analysis of the local extremum values.

The experience showed the high efficiency of the analytical search methods when the estimation of the secondary optimization functional gradient is performed by the current output and intermediate signals.

The main problems considered in the field of the secondary optimization functional extremum search are related to the fact that this functional is multi-extremal and it exists in a rather multi-dimensional space of the neural network adjustment coefficients.

This is the reason that the modern neural network methods of the secondary optimization functional extremum search require the development in different directions, and some of them are described below.

17.6.9.1
The Control of Parameters in the Extremum Search Procedure for the Multi-Extremum Secondary Optimization Functional

The gradient local extremum search procedure for the multi-extremum optimization functional is a very important element of the multilayer neural network adjustment algorithm. In the simplest case, the weight coefficient K^* at the functional gradient is determined in the empirical way in the process of solution of each specific task and is left constant in the adjustment procedure. Since the 1960s, researches have tried to make coefficient K^* be variable (decreasing) with time during the adjustment procedure. This was done in order to decrease the adjustment error in the steady state. But this led to the significantly sharp increase in the adjustment duration (in the transient process of the multilayer neural network adjustment). At present, a considerable part of this problem remains open. Some efforts are made to control the value of the coefficient K^* by the current error value and by the gradient functional.

17.6.9.2
The Modifications of the Global Extremum Search Algorithms for the Multi-Extremal Secondary Optimization Functional

The secondary optimization functional in the multilayer neural network is multi-extremal by definition. The reasons for this are the following:

- The input signal is rather complex (for example, the distribution of patterns aggregate in the multi-dimensional space of indications at the patterns' recognition task solution is multi-modal);

- The variants of the task solution are multivariate;
- The open neural network structure is flexible.

The search methods for the global extremum (or, in addition, for several local extrema) are at present only in the stage of formation.

One of such methods related to the set of ejections of the random initial conditions in the space of the neural network weight coefficients and to the search of the global extremum for the secondary optimization functional was presented and investigated in [17-2]. The convergence of the presented procedure by the number of initial conditions under the fixed number of local extrema was shown there. Some methods of this procedure resulting in the time decrease for the global extremum search are known. The method of "annealing" is an example.

The method of "annealing" can be used in the adjustment process in the following way. The independent variables (the neuron weights in the case of the neural network) undergo random changes. The values of the minimized neural network's functional are stored for each changed set of variables, and then the best set is selected. A relatively large range of the random-value generator that changes the neuron weights is taken at the beginning of the process. The set of the variables' values (weights) corresponding to the best functional value is then selected after several changes. And this set of variables is then taken as the initial one for the following procedure of random changes but with the decreased range of the random-value generator.

The gradient algorithm is effective in finding the local minimums in the case of neural network weight adjustment during the learning procedure. In general, the mixture of the "annealing" and gradient methods is the most effective algorithm. First, the "annealing" algorithm is used to find the initial weights. The gradient descent algorithm is used thereafter to bring the system to the nearest local minimum. Then the "annealing" algorithm is used at this point again in order to leave this local minimum. These stages are repeated until one has the possibility to leave the recurrent local minimum. In the latter case, it can be considered that the global minimum is obtained.

17.6.9.3
Filtering and Extrapolation of the Signal corresponding to the Estimation of the Secondary Optimization Functional Gradient

As a rule, the decision to change the weight coefficients in the known neural network adaptation algorithms is taken in each operation cycle according to the results of one single pattern passing through the network. The experience of using the filter with the memory $m_n \neq 1$ [17-2] in the adjustment circuit showed the increase in the adjustment effectiveness for stationary and non-stationary patterns at the multilayer neural network input. Some attempts to speed up the learning process through the application of the weight coefficients' extrapolation procedure during the neural network adjustment are known from the literature. The filter synthesis in the weight coefficients' adjustment circuit is poorly investigated though it is a perspective algorithm in the general procedure of the multilayer neural network synthesis.

The following parameters must be determined in the process of the multilayer neural network adjustment algorithm:

- The filter memory;
- The filter type according to the a priori selected form of the neural network input signal nonstationarity.

17.6.9.4
The Multilayer Neural Network Adaptation Algorithms with the Adjustment of the Coefficients for the "Slope" of the Activation Function

When using the activation function with a variable "slope," the separate neuron in the neural network is described by the expression

$$y = \frac{2}{\pi}\operatorname{arctg} B\sum_{i=0}^{w} a_i x_i = \frac{2}{\pi}\operatorname{arctg}\sum_{i=0}^{w}(Ba_i)x_i$$

It is seen therefore that there is no sense in organizing the adjustment circuit for the coefficients B and a_i at the construction of adaptation algorithms. In the neural network consisting of the neuron set that contains a separate neuron subset or all the sets with the activation function of the same "slope," the organization of the adjustment circuit for the coefficient B, as well as for the separate coefficients, is necessary in order to decrease the whole adjustment duration, i.e., to decrease the time required for the task solution.

17.6.9.5
About the Use of the Second Derivative of the Secondary Optimization Functional

The multilayer neural network adjustment algorithms with the use of the second derivative of the secondary optimization functional were developed at the end of the 1960s. The works of the 1990s did not contribute significantly in this field. However, at the beginning of the 1970s, the experiments using the second derivatives showed that the noise level in this case is very high and the use of the second derivative becomes inefficient. At present, this situation is the same.

17.6.9.6
Selection of the Initial Conditions for the Gradient Procedure of the Extremum Search of the Secondary Optimization Functional

The choice of the initial weight coefficients of the adaptive neural network is an important condition to speed up the task solution procedure. Therefore, from our point of view, the widespread approach to choose the zero values for the weight coefficients or the random values with the uniform distribution in the given turn-down as the initial conditions is incorrect.

Even during the solution of the problem of recognition of two pattern classes, it was clear that the initial weight coefficient values must be selected by generating the dividing surface configuration, implemented by the neural network, in the form of a multidimensional "chess-board" with the uniform distribution of "black" and "white" squares. Each color corresponds to the first and second classes of patterns in the physically implemented multidimensional space of indications [17-2, 17-3]. Such a multidimensional "chess-board" is formed by the hyperplanes corresponding to the neurons of the first layer.

The initial weight coefficients of the adaptive neural network can also be chosen as the weight coefficients of the neural network with variable structure after its learning stage termination. As it was mentioned above, this is possible only for the pattern recognition solution and for clasterization (self-learning) with the developed adjustment algorithms for the multilayer neural networks with flexible structure.

The problem of function approximation (extrapolation) is an example of the effective solution of the problem of the initial weight coefficients' selection for the multilayer neural network adjustment. In this case, assuming that the neural network is an effective nonlinear filter (extrapolator), it is expedient to choose the weight coefficients of the neural network implementing the equivalent nonlinear filter or extrapolator as the initial conditions.

Hence, the choice of the initial conditions for the multilayer neural network adjustment possesses the following properties:

- It is specific for each specific task that is solved by the neural network;
- It is aimed at the acceleration of the adjustment process (and therefore at the acceleration of the task solution) by putting the neural network into the domain of the global extremum of the secondary optimization functional;
- As a result, it allows one to increase the equivalent ratio between productivity and the cost during the specific task solution.

The examples of the particular solutions for the problem of the initial condition choice are given in [17-7].

17.6.10
Formation of the Adaptation Algorithms in the Multilayer Neural Networks

The base for the formation of the adjustment (adaptation) algorithms in the multilayer neural networks includes

- The analytical expression for the secondary optimization functional and the analytical expression for its first derivative or the estimation of the sign of the first derivative;
- The analytical expression for the secondary optimization functional extremum search with the use of its first derivative expressed through the input and output signals of the adjusted multilayer neural network.

17.7
Verification of the Adjusted Multilayer Neural Network

The elaboration of the special test system for the neural networks with different structures is an important element for the increase in the reliability of prospective neurocomputers. The development of such tests for the adjusted neural network is a branch of the perspective investigations in the neural network theory domain. This section relates both to the neural networks with variable structure as well as to those with fixed structure.

The elaboration of the typical neural network input signal classes is necessary for the objective test of the adaptive neural network performance quality. Furthermore, the system of tests is always specific for the specific solution task.

The class of signals with Laplace transformation

$$\left\{1,\ \frac{1}{s},\ \frac{1}{s^2},\ \frac{1}{s^3},\ \ldots\right\}$$

is a typical example of the neural network input signal class. In this case, the test of the control system performance quality is carried out by the feed of the corresponding typical signals (δ-function, unit step, linear signal, etc.) to the system input with the following analysis of the transient process and of the error in the terminal steady state (the order of the control system astaticism).

The typical neural network input signal class must always possess some parameter characterizing the complexity of the solution task. This parameter is apparent in the aforementioned example. As far back as the beginning of the 1960s, for the pattern recognition tasks oriented onto the performance with the random samples of complex unknown multi-modal distributions, the random samples of multi-modal distributions were suggested as typical neural network input signals. Hence, the distribution modes represented a normal distribution and the mode centers were situated along the hyperbisector of the multidimensional space of indications alternating for each class [17-2, 17-3]. Two parameters were taken as the solution task complexity indexes, namely the number of the distribution modes and the variance of each separate mode. The indications of the modes belonging to the specific class in the case of a self-learning neural network performance mode were absent.

The selection of the typical neural network input signal class is an important task for the researcher who desires to demonstrate, more or less objectively, the advantages of his neural network algorithm elaborated for the solution of the specific formulated task. The typical neural network input signal class must always possess some parameter characterizing the complexity of the solution task. This parameter is apparent in the aforementioned example. As far back as the beginning of the 1960s, for the pattern recognition tasks oriented onto the performance with the random samples of complex unknown multi-modal distributions, the random samples of multi-modal distributions were suggested as typical neural network input signals. Hence the distribution modes represented a normal distribution, and the mode centers were situated along the hyperbisector of the multidimensional space of indications alternating for each class [17-2, 17-3]. Two parameters were taken as the solution task complexity indexes, namely the number of the distribution modes and the variance of each separate mode. The indications of the modes belonging to the specific class in the case of a self-learning neural network performance mode were absent.

The selection of the typical neural network input signal class is an important task for the researcher who desires to demonstrate, more or less objectively, the advantages of his neural network algorithm elaborated for the solution of the specific formulated task.

17.8 Elaboration of the Plan of Experiments

All the undefined parameters of the neural network and the input and output test signals must be taken into account during the elaboration of the plan of experiments. These parameters must be ordered and represented in the form of the experimental plan with the elaborated neural network solution algorithm.

The problem of the small-size sample of the input signal presented to the neural network for its training must be considered separately. This problem is a significant problem in the large number of information processes aimed at decision making. This problem was also significant in the process of standard statistical decision making. However, in this case, the strict limitations related to the small-size sample made it necessary to take into account the a priori information concerning the distribution function of the processed signals. The neural network methods of the information processing were elaborated namely because this a priori information is absent and therefore the distribution functions can be very different, complex or unknown.

The problem of the small-size sample is very significant in the following two cases:

1. When the number of measurements is small and cannot be increased in principle;
2. When the number of measurements is increasing but the decision must be made as far as the measurement results are obtained.

We present below the part of the technique for planning of experiments. This part deals with the specific multilayer neural network performance under the conditions of the small-size learning sample for the adjustment of the multilayer neural network with fixed structure:

1. The procedure of multiple repetition of the sample with a relatively small size on the input of the neural network is an effective technique to increase the adequacy of decision making by the multilayer neural network. *The adequacy of decision* means the estimation of the correct recognition probability, the mean-square error of the function approximation, or any other evaluation depending on the task that the multilayer neural network solves;
2. One of the possible ways the neural network is implemented for the property of generalization by the similarity is the artificial generation of additional samples on the neural network input. Moreover, the additional samples must possess the mathematical expectation in the form of initial small-size sample components, and they must have a different variance. With that, the value of the variance may change in the process of execution in the plan of experiments;
3. The initial conditions' selection at the multilayer neural network adjustment is a very important problem that actively influences the speed of computations in the neural network logical basis (the speed of the adjustment algorithms' convergence with one of the local extrema or with the global extremum of the optimization functional) and the quality of the task solution. Under the different methods of the initial conditions' selection, each time different adjustment results will be obtained. Averaging across the results will give thereafter, at the limited specified sample, additional information about the neural network performance quality. In this case, the initial conditions can be taken either randomly (by the initial weight coefficient generation through the random-value generator) or by calculation across the limited sample with the help of the adjustment algorithms with variable structure;
4. The division of the initial sample into the smaller samples, the multilayer neural network adjustment with the use of technique mentioned in points 2 and 3, and the results' averaging across the set of the mentioned smaller parts of the initial sample

can be regarded as an additional method to analyze the neural network generalization properties;

5. The resultant stage of the suggested technique is the stage of the averaging of the adjustment results across the set of variants mentioned in points 2, 3 and 4. The obtained distribution function for the quality index will have some mathematical expectation and variance. The mathematical expectation of the quality index is the main characteristic of the multilayer neural network operation with the sample of the ultimate object. The value of the quality index variance is the evaluation of its uncertainty. At the high value of the obtained variance, one must undertake some efforts that would result in the improvement of the solution quality. One of such possible efforts consists in the increase of the structure complexity of the multilayer neural network.

One can use methods mentioned in points 2 and 4 when operating with the small-size sample in the case of the multilayer neural network with flexible structure.

The presented technique is an illustration of the possibility to partially compensate the shortage of the experimental information by the additional computational resource. This technique can be used not only for the recognition pattern or function approximation task solutions under the relatively small number of the experimental observations but also for the general task solutions.

17.9 About the Importance of the Unification of Designations in the Process of Synthesis of the Neural Network Adjustment Algorithms

The investigations in this field show that the understanding of the essence of the studies and of the peculiarities in the different algorithmic approaches will be much more transparent in the case of some unification of designations in the scientific literature. We present below some version of such a unification.

Designations

x – the neural network input signal;
N – the dimensionality of the feature space;
w – the neural network weight coefficient;
i – the number of the feature ($i = 0, \ldots, N$);
g – the analogous output signal of the neuron (the input signal of the unit implementing the activation function after the multi-input summation unit);
y – the output signal of the neuron or the neural network;
K – the number of classes;
K_p – the number of solutions;
$f(g)$ – the activation function;
H_1 – the number of neurons in the first layer;
W – the number of layers in the multilayer neural network;
H_w – ($w = 1, \ldots, W$) – the number of neurons in the m-th layer of the multilayer neural network.

17.10 About Myths in Neural Network Theory

Fuzzy logic is one of the bases of neural network theory development. The neural network is one of the most efficient methods for the implementation of the fuzzy logic concept.

Along with that, different scientific fields emerge in neural network theory development. But the detailed analysis shows that these fields are only some narrow and particular case of the separate aspects of neural network theory. Therefore, we propose discussing some of the definitions, such as

- Genetic algorithms;
- Support vector machines;
- Wavelet networks;
- RBF-networks;
- Principle component analysis;
- Evolutionary programming.

The attempts to pull out some parts of the neural network theory and to make them independent only weaken these parts. The aforementioned list shows the examples of such neural network theory divisions.

The same particular interpretations of the neural network theory are the classic methods of the mathematical statistics and methods of the potential functions that were actively discussed at the end of the 1960s and the beginning of the 1970s.

The main idea consists in the proposition to transform the earlier suggestions concerning the "emotional" definition of the algorithm and the similar suggestions that will be made in the future for some vector of **quantitative parameters** with the corresponding quantitative explanation of why the new algorithm changes this or that parameter.

Such a qualitative description can be made for the various known neural network types:

- Kohonen networks;
- Elman networks;
- Hopfield networks;
- ART neural network, etc.

In this case, the **quantitative** limitation of the different neural networks will be immediately seen for their users.

17.11 Conclusion

In this section, we represent the most optimal, from our point of view, design cycle for the neural network solution algorithms that can be implemented at present.

The number of investigations in the domain of the neural network theory is increasing. This strengthens the requirements for the comparison and the detailed classification of the different neural network synthesis algorithms. It must be done by means

of the comparison of the a priori information on the neural network synthesis in each particular case.

1. A priori characteristics of the neural network "supervisor instruction" space, i.e., the number of the pattern classes (two, K, continuum);
2. A priori characteristics of the neural network input signal nonstationarity;
3. The two-argument function of the "supervisor qualification" of the neural network. The arguments are the indexes of the corresponding classes;
4. The function of the "supervisor's own opinion" about its abilities. This is also the two-argument function with the arguments that are the indexes of the corresponding classes;
5. A priori probabilities of the classes' emergence;
6. A priori characteristics of the neural network solution space (two, K_p, continuum of solutions);
7. The class of criteria for the primary neural network optimization;
8. The function of losses that emerges when one pattern system is erroneously taken as the pattern system belonging to another class;
9. A priori information about the conditional distribution function $f'(x/\varepsilon)$;
10. A priori information about the fixed structure of the open neural network during the development of the neural network with fixed structure that is tuned through the closed-cycle procedure;
11. A priori information about the class of structures during the development of the neural network with flexible structure;
12. A priori information about the difference between the primary and secondary optimization functionals during the development of the neural network with fixed structure that is tuned through the closed-cycle procedure;
13. A priori information about the method of searching of the secondary optimization functional extremum;
14. A priori information about the limitations on the adjustment coefficients;
15. A priori information about the procedure of selection of the elements of the parametric matrix K^* of the search system for the secondary optimization functional extremum;
16. A priori information about the search oscillation parameters in the case of when the neural network adaptation algorithm cannot be designed in the analytical form;
17. A priori information about the initial conditions for the adjustment procedure;
18. A priori information about the class of the typical neural network input signals;
19. A priori information about the degree of complication of the open neural network structure on each iteration step and about the form of this complication.

The objective comparison between the multilayer neural networks of different types must be performed through the comparison of the a priori information about their design and the comparison of their performance quality with the typical and real input signals.

Table 17.1 shows the comparison of the neural network synthesis procedures described in [17-2, 17-3] and in the large number of American studies concerning the error back-propagation methods.

Table 17.1. Comparison of the neural network adjustment methods

No.	The characteristics of the neural network synthesis procedure	Russian adaptation methods for the multilayer neural networks	The error back-propagation method	Comments
1	The date of the study and publication	1965–1971, 1970–1974	1976–1984	
Characteristics of the input signals				
2	The number of pattern classes (the graduation level for the signal of the supervisor instructions about the input pattern belonging to the different classes)	2, K, continuum	2	
3	The characteristics of the transient properties of the input patterns as the random signals	Stationary, nonstationary	Stationary	
4	The "supervisor qualification" characteristics	Arbitrary	Learning ($B = 1$) Rare Self-learning ($B = 0$)	
5	The supervisor's opinion about his abilities	+	–	
6	A priori probability of the emergence of pattern classes	Arbitrary	Equal	
The solution space characteristics				
7	The number of solutions	2, K, continuum	2	For any number of classes
8	A priori information about the conditional density of probability distribution for the pattern classes	Can be considered	Not considered	
The primary optimization criteria				
9	The class of the primary optimization criteria	The average risk function without and with consideration of the constraints on the components for different classes, maximum of the a-posteriori information, etc.	Energy function, mean-root square error	Russian technique: – min R (average risk function) – min R with $p_1 r_1 = \text{const}$. (the component of the average risk function) – min R with $p_1 r_1 = p_2 r_2$, etc.
10	The loss matrix (function)	Arbitrary	Symmetrical	

Table 17.1. *Continued*

No.	The characteristics of the neural network synthesis procedure	Russian adaptation methods for the multilayer neural networks	The error back-propagation method	Comments
The multilayer neural network structure				
11	Types of multilayer neural network structures	Multilayer neural network with complete and incomplete sequential, cross and feedback connections. Arbitrary structure adequate to the solved problems	The three-layer neural networks with complete sequential connections; Particular structures with the feedback coupling	
The secondary optimization functional				
12	The method of the secondary optimization functional selection corresponding to the primary optimization functional	+	–	
The search methods of the secondary optimization functional extremum				
13	Combined methods (gradient and random)	+	–	
14	Method of stochastic approximation	+	–	
15	Consideration of the information about the constraints on the adjustment coefficients (for example, the value of the change or the rate of the change)	+	–	
16	The possibility of using the search oscillations	+	–	
17	The possibility of filtering in the adaptation circuit for the estimation of the secondary optimization functional gradient	+	–	
18	The initial condition selection in the adjustment coefficient circuit	+	–	
Typical input signals				
19	The typical input signal selection	+	–	

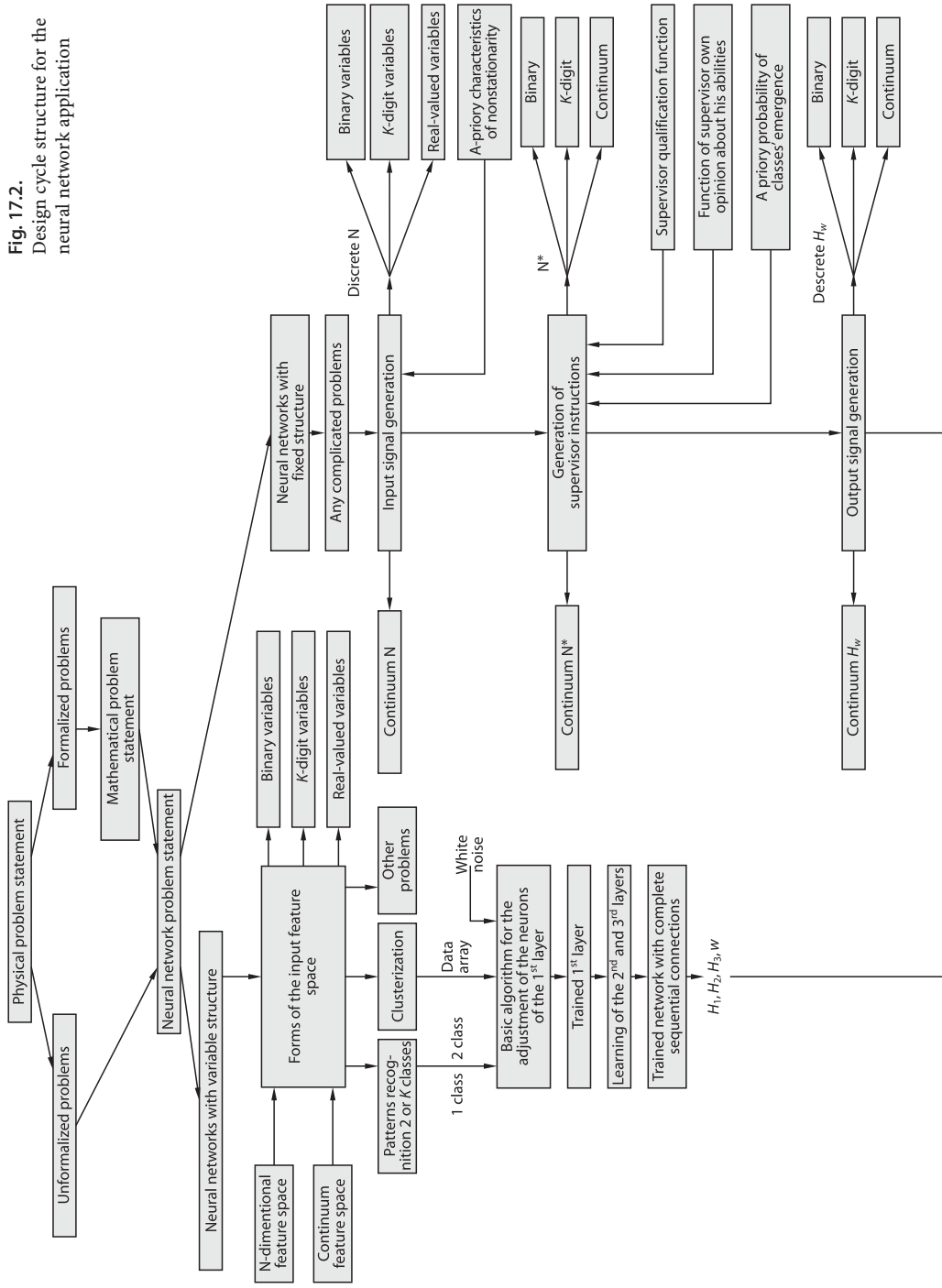


Fig. 17.2.
Design cycle structure for the neural network application

Criterion form
Loss function
Generation of the primary optimization criterion
Generation of the open neural network structure
Neural networks with complete sequential connections
Neural networks with complete feedback connections
Other structures with explanation of their architecture
Selection of activ.func. in the layers except the last one
Selection of neurons number in the layers except the last
Selection of the number of neurons in the layer
w
H1,...HW-1
f1,...fW-1
Algorithm and program for the generation of the secondary optimization functional and its derivatives by the signals in the system
Selection of the algorithm for the search of the extremum of the secondary optimization functional (SOF)
Constraints on the weighting coefficients
Gradient algorithms
Variant 1
Variant 2
Variant 3
Variants with the use of the 1st derivative of SOF
The use of the 2nd derivative of SOF
Selection of the initial conditions (IC)
Zero IC
Random IC
IC adequate to the solved problem
Algorithms with the use of the search oscillations
Algorithms of the random search
Combined searching algorithms
Generation of the adjustment algorithm
Tests for the performance quality of the trained neural network
Plan of experiments under the small sample of the input data
Plan of experiments

Figure 17.2 represents the structure of the neural network synthesis for the solution algorithms. This structure was developed for several years. On the one hand, it includes the results of the large number of investigations that are represented by their authors as "a new method…" or "original approach…". However, these approaches are rather particular. On the other hand, this structure is incomplete and represents only the additional line in the neural network theory that might not be known to the author.

However, the author does not consider each new study that comes in his sight, from the point of view reflected in this figure. And he asks himself if this new study might be the particular case of some already known works. This was the reason for writing the aforementioned section "About Myths in the Neural Network Theory". At any rate, the author conceives all the works that further develop the structure shown in Fig. 17.2 with great satisfaction.

When considering the neural network solution algorithms, the problem of a so-called "false" statement of the mathematical problem often arises. Let us explain it in some examples. The solution of the systems of the linear algebraic equations is sometimes considered as the problem of the matrix inversion in the classical mathematic approach. However, the neural network algorithms for the solution of these two problems are very different. And the problem of the matrix inversion becomes not necessary, i.e., this problem becomes "false". A similar situation is observed in the problem of solution of the systems of the ordinary nonlinear differential equations. These equations are the formalized description of the behavior of physical objects. The use of the neurocomputers eliminates both the necessity of the formalized description of the physical objects as well as the necessity of the solution of the systems of the ordinary nonlinear differential equations. In this case, the formalized description of the physical objects is performed by the use of the neural networks of different structures. Namely due to this reason, the neural network control decreases significantly the interest in the solution of the systems of the ordinary nonlinear differential equations.

Neuromathematics establishes some new problems for computational mathematics that either were not solved before or were solved insufficiently. Some of these problems are the following:

- The initial condition selection;
- The universalization of the different problem solution algorithms;
- The selection of the classes of typical input signals for the test (verification) of the neural network performance quality;
- The selection and rejection of the "false" problem statements;
- The investigation and dynamic control of the solution procedure;
- The problem of the number of solutions related to the multi-extremum properties of the optimization functional.

The development of the neural network solution algorithms allows for the efficient selection of the initial conditions as well as for the use of several methods for the dynamic control of the problem solving procedure including the control of the rate of convergence:

- The control of parameters (the modification selection) of the iteration procedure for the multi-extremum optimization functional search in the problem solving process;
- The filtering and extrapolation of the signal corresponding to the estimation of the optimization functional gradient in the problem solving process;
- The adjustment of the multilayer neural network's activation function slope;
- The selection and analysis of the special structures of the neural networks adequate to the class of the solved problems (cellular-like or continuum neural networks, neural networks with lateral, random and feedback coupling, neural networks with flexible structure).

The problem of the number of solutions in the algorithms development is very important. In the case of the neural network algorithms, this problem is related to the multi-extremum characteristics of the optimization functional and to the possibility of the formation of the multi-extremum functional through the variation of the open neural network structure.

The neural network theory at the present time represents an independent field of science. The main prospective lines of the neural network theory relate to the solution of the complex practical problems. Some of them are the following:

- The continuum neural networks with the formal consideration of the continuum number of the input signals, of the input channels or of the neurons in the layers;
- The investigation of the structural or parametric reliability of the neural networks used in the neurocomputer implementation technology;
- The parallelizing of the neural network algorithms for the different types of the commutation kernels in the super-neurocomputers;
- The neural networks that provide the invariance to the group of transformations (for example, the scale-invariance or signal-invariance, Lorentz invariance, etc.);
- The analytical description of the neural networks with continuum adaptation using the apparatus of the linear sequential Gill machines, etc.

Unfortunately, perhaps due to poor awareness, a large number of "home-bred" neural network algorithms emerge. The achievement of the first positive results on the basis of these algorithms can provoke an illusion about the "completeness" of the neural network theory. However, this theory is only in the initial phase of its development. Evidently, in the present work, the entire list of neural network theory problems is not all enumerated. The gradual progress in this domain must improve the solution of the vast number of existent problems as well as pose new problems. We consider that the neural networks will be the foremost essential tool for the investigation of complex problems of the modern world.

The present study was performed in the framework of the state agreement with the Federal Agency for Science and Innovations in the development works No. 02.435.11.1003.

References

[17-1] Galushkin AI (1970) Multilayer systems of pattern recognition, Moscow: MIEM
[17-2] Galushkin AI (1974) Synthesis of multilayer systems for pattern recognition. Moscow: "Energiya"
[17-3] Galushkin AI (2000) Neural network theory, Moscow: IPRGR
[17-4] Neuromathematics (2002) V. 6. series "Neurocomputers and their application". Moscow: Radiotechnika
[17-5] Galushkin AI (2003) Neuromathematics (problems of development). Neurocomputer 1
[17-6] Galushkin AI (2004) Neurocomputers and their application in China on the boundary of millenniums. Moscow: Goryatchaya liniya – Telekom
[17-7] Galushkin AI (2005) Generation of the initial conditions for the fastening of neural network coefficients tuning in the optimization tasks. Neurocomputer 5

Conclusion

Neural network theory is an independent branch of science at present. The main perspective lines of the neural network theory relate to the solution of complex practical problems. The following fundamental problems can be mentioned:

- Continuum neural networks with a formally considered continuum of input channels, neurons in the layers, etc.;
- Neural network reliability;
- Neural networks providing the invariance to the group of transformations (for example, to the shift, rotation, patterns or signal scaling);
- Analytical description of neural networks with adaptation circuits using technique of Gill linear sequential machines, etc.

The number of scientific investigations in the field of neural network theory is increasing. That is why the analytical approach is required for detailed classification of different methods for the neural network synthesis problem solutions. And the most important domain for the application of such approaches is the selection of the a priori information required for the multilayer neural network synthesis in each particular case.

1. A priori characteristics of the neural network teacher instruction space – the number of pattern classes (2, K, continuum);
2. A priori nonstationary characteristics of the neural network input signal;
3. Neural network teacher qualification function of two arguments representing the indexes of corresponding classes;
4. Function of "teacher's slant about his capabilities" of the neural network. This is also a function of two arguments representing the indexes of corresponding classes;
5. A priori probabilities of classes' appearance;
6. A priori characteristics of the neural network solution space (2, K_p, continuum);
7. Class of the neural network primary optimization criteria;
8. Function of losses that take place when the system considers a pattern to belong to a wrong class;
9. A priori information about conditional distribution functions $f'(\mathbf{x}/\varepsilon)$;
10. A priori information about the fixed structure of the open-loop neural network in the design of the neural network with fixed structure adjustable in the closed cycle;
11. A priori information about the structure class in the design of the neural network with flexible structure;

12. A priori information about the distinguishing features of the primary and secondary functionals in the design of the neural network with fixed structure adjustable in the closed cycle;
13. A priori information about a search method for the secondary optimization functional;
14. A priori information about the presence and form of the constraints imposed upon the adjustable coefficients;
15. A priori information about a selection method for the coefficients of the parametric matrix K^* of the secondary optimization functional extremum search system;
16. A priori information about the search oscillation parameters in the case of when the neural network adaptation algorithm cannot be designed in the form of the analytical system;
17. A priori information about the initial conditions for the adjustment procedure;
18. A priori information about the class of the neural network's typical input signals;
19. A priori information about the degree of the open-loop neural network structure complication at each step and about a method of such a complication realization;

The objective comparison between the multilayer neural networks of different types must be performed on the basis of comparison of the available a priori information for their design and their operation quality at the typical and real input signals.

Literature

1. Solodovnikov VV (ed) (1965) Analytical self-adaptive systems. Moscow: Mashinistrojenije, p 355
2. Rosenblatt F (1965) Principles of neurodynamics. Moscow: Mir, 480 p
3. Galushkin AI (1970) Multilayer pattern recognition systems. Moscow: MIEM, p 167
4. Galushkin AI, Yumashev SG (1970) Use of piecewise-linear divisional surfaces in pattern recognition problems. MIEM proceedings 6:238–254
5. Galushkin AI (1974) Synthesis of multilayer pattern recognition systems. Moscow: Energiya, 367 p
6. Galushkin AI, Shmid AV (1992) Structure optimization of multilayer neural networks with cross connections. RNNS/IEEE Symposium, p 509–520
7. Galushkin AI, Shmid AV (1992) Structure optimization of multilayer neural networks with cross connections. Neurocomputer 2:7–11
8. Galushkin AI (1977) Continuum models of multilayer systems for pattern recognition. Automation and computer technology 2:43–48
9. Galushkin AI (1992) Continual neural networks. RNNS/IEEE Symposium, p 1056–1067
10. Galushkin AI (1992) Continual neural networks. Neurocomputer 2:9–14
11. Galushkin AI (1993) Continual neural network. Int. Joint Conf. on neural network, IJCNN'93, Nagoya, Japan, p 345–398
12. Galushkin AI (1970) Unified approach to the problem of learning and self-learning for the pattern recognition systems. MIEM proc. 6:104–120
13. Galushkin AI (1972) Selection of the primary optimization criteria and optimal model of the pattern recognition system for *K* classes in the learning mode. In: Automatic control and computer facilities (pattern recognition). Mashgiz 10:101–115
14. Galushkin AI, Zak LS, Tiukhov BP (n.y.) To the comparison of optimization criteria for the adaptive pattern recognition systems. Kiev: Kibernetika 6:122–130
15. Galushkin AI (1971) Implementation of the primary optimization criteria in the pattern recognition systems adjustable by the closed-cycle in the learning mode. MIEM proc. 23:191–203
16. Galushkin AI (1970) Analysis of one extremum search iteration method. Riga: Automatics and computer engineering 2:38–40
17. Tsypkin Ya Z (1968) Adaptation and learning in automatic systems. Moscow: Nauka, p 399
18. Tsypkin Ya Z (1970) Learning system theory foundation. Moscow: Nauka, p 251
19. Galushkin AI, Shmid AV (1971) Iteration methods for the search of extremum of multivariable functions under equality-type constraints. Riga, Automatics and computer engineering 4:88–91
20. Galushkin AI, Tiukhov BP, Chigrinov VG (1971) About convergence of one random search method for the search of local and global extrema of a multivariable functions. MIEM Proc. 23:205–209
21. Rastrigin LA (1968) Statistical search methods. Moscow: Nauka, p 376
22. Galushkin AI (1973) About adaptation algorithms in multilayer pattern recognition systems. Dokl. AN USSR
23. Galushkin AI (1970) Methods of pattern recognition system synthesis. MIEM proc. 6:133–171
24. Victorov NV, Galushkin AI (1976) Design and investigation of pattern recognition systems under the arbitrary teacher qualification. Medical radio-electronics, pp 96–106
25. Vanyushin VA, Galushkin AI (1972) Design and investigation of multilayer pattern recognition systems. In: Acad. Berg AI (ed) Some problems of biological cybernetics. Leningrad: Nauka

26. Galushkin AI, Vasilkova TA, Slobodeniuk VA, Tyukhov BP (1971) Analysis of the nonstationary pattern recognition system dynamics. Proc. MIEM 23:210–227
27. Victorov NV, Galushkin AI (1976) Design and investigation of pattern recognition systems under the arbitrary teacher qualification. Medical radio-electronics, VNII Mejdounarodnoy tekhniki
28. Galushkin AI, Koudryavtsev AM (1976) Matrix inversion with the help of a multilayer system based on the threshold elements. Kiev, Naukova Dumka: Cybernetics and computer facilities 33
29. Galushkin AI, et al. (1991) The pattern recognition system with variable structure on the base of the personal computer and transputer system. SERC/DTI Transputer Initiative Mailshot, p 56–73
30. Grachev LV, Simorov SN (1992) Statistical research into multilayer neural network. RNNS/IEEE Symp., p 1172–1178
31. Grachev LV, Reznitsky IV (1992) Synthesis of recognition systems with variable structure by three-layer neural networks. Simorov S.N. Statistical research into multilayer neural networks. RNNS/IEEE Symp., p 1086–1097
32. Gerasimova AV, Grachev LV (1992) Representativeness of learning samples for paradigm of variable-structure neural networks. RNNS/IEEE Symp., p 449–456
33. Gerasimova AV, Grachev LV (1992) To the problem of the learning sample representativity for the paradigm of the neural networks with flexible structure. Neurocomputer 3,4:3–6
34. Fomin Yu I, Galushkin AI (1980) Methods of technology diagnostics of threshold element networks. Tekhnika sredstv sviazi, Sistemy sviazy, 2:84–94
35. Fomin Yu I, Galushkin AI (1981) Methods of failures parallel diagnostics in threshold elements networks. Kiev, Naukova Dumka: Elektronnoye modelirovaniye 3:89–92
36. Galushkin AI, Fomin Yu I (1979) About optimality of restoring organs realizing majority voting. Tekhnika sredstv sviazi, ser. ASU 3:56–61
37. Fomin Yu I, Galushkin AI (1982) Majority voting and restoring organs for its implementation. Kiev, Naukova Dumka: Cybernetics and computer facilities 55:91–97
38. Gill A (1974) Linear sequential machines. Moscow: Mir, p 287
39. Faradjev RG (1975) Linear sequential machines. Moscow: Sov. Radio
40. Galushkin AI (1996) Summary and perspectives of the multilayer neural network theory development (1965–1995). In: Proceedings of the Neurocomputer scientific center. Moscow

Author's Publications on Neural Network Theory

1. Galushkin AI (1965) Calculation of the optimal integrating filters. Izvestia VUZov, Electromechanics Series No. 2
2. Galushkin AI (1966) Some problems of the optimal digital filter designing. Questions of Radio-electronics, Technics in general Series No. 21
3. Galushkin AI (1967) Method of the logarithmical frequency characteristic construction. Collection "World in a young scientist's opinion". First Moscow conference of young scientists. Science
4. Galushkin AI (1968) Method of the frequency characteristic construction. Machgiz: Collection "Automatic management and computing machinery", No. 8
5. Galushkin AI (1968) Some problems of the theory and the realization of optimal discrete filters. In: Solodovnikov VV (ed) Collection Automatic management and computing machinery, No. 9, Machgiz
6. Galushkin AI (1968) Equipment for electric signal recognition. Certificate of recognition No. 235420 dated 01.02. 1968
7. Galushkin AI (1968) Pattern recognition. Radio-electronics 1967. Review over the foreign press materials, NIIEIR
8. Galushkin AI (1969) Introduction to the mathematic model of septron. Seminar Works for informational methods in the systems of management, dimension and control, Vladivostok
9. Galushkin AI (1969) The synthesis of pattern recognition systems. Russian seminar on adaptive systems, Russia, Leningrad
10. Galushkin AI (1969) About input signal characteristics on pattern recognition systems. The report theses of the Russian conference on Technical cybernetic, Russia, Minsk
11. Galushkin AI (1969) About piecewise-linear separating surfaces in problems of pattern recognition. The reports theses of I. Russian interuniversity conference on technical cybernetics, Moscow, (in co-authorship)
12. Galushkin AI (1969) Synthesis of non-stationary pattern recognition systems trained on an opened cycle. Works MIEM, issue 6
13. Galushkin AI (1970) Prediction of individual safety of electronic and semiconductor instruments using the methods of pattern recognition theory. Works MIEM, issue 10
14. Galushkin AI (1970) Problems of using the pattern recognition systems for management and optimization of technological processes of production and projecting electronic plans. Works II All-Union seminar for High school workers "Construction and industry of radio-technique", Novgorod
15. Galushkin AI (1970) Electric signal recognition system on fiber optics with two orthogonal fixed piezo-transformers. Certificate of recognition No. 1354081, 18–24 (att.)
16. Galushkin AI (1970) The common approach to construction of optimum models of pattern recognition systems in a training mode. Works MIEM, issue 12, Editor: Armenski EV
17. Galushkin AI (1970) About non-stationary pattern recognition systems. Works MIEM, issue 12, Editor: Armenski EV, (in co-authorship)
18. Galushkin AI (1970) Introduction to the mathematical model of signal recognition on septrons. MIEM Monography
19. Galushkin AI (1970) Multilayered pattern recognition systems. Works MIEM, monograph
20. Galushkin AI (1970) Determination of intervals between signals with septron assistance. Materials of IV Republic Scientific Conference for young researchers for system technics, vol. 3, IK AN USSR, Kiev

21. Galushkin AI (1970) Iterative methods of search of an extremum: multivariable functions under constraints of equality type. Materials of IV Republic Scientific Conference for young researchers for system machinery, vol. 3, IK AN USSR, Kiev
22. Galushkin AI (1971) Synthesis of non-stationary pattern recognition systems trained on an opened cycle. Technical cybernetics Izv. AN USSR No. 1
23. Galushkin AI (1971) About the arrangement of objects of geometric recognition. Works MIEM, Some questions of cybernetic system theory, issue 14
24. Galushkin AI (1971) Moment approach to determinate tasks of self-learning pattern recognition systems. Works MIEM, Some questions of cybernetic system theory, issue 14
25. Galushkin AI (1971) Arrangement of recognition systems of signal groups on septrons. Works MIEM, Some questions of cybernetic system theory, issue 14
26. Galushkin AI (1971) About signal recognition systems on septrons with cut photo elements. Works MIEM, Some questions of cybernetic system theory, issue 14
27. Galushkin AI (1971) About pattern recognition system adaptation for control and optimization of technological processes of industry of semiconductors. Works MIEM, Some questions of cybernetic systems theory, issue 14
28. Galushkin AI (1971) Research of technological diffusion operation with pattern recognition theory methods. Works MIEM, Some questions of cybernetic system theory, issue 14
29. Galushkin AI (1971) Adaptation of learning algorithms of K-class pattern recognition systems for the analysis analyze of technological operation sputtering. Works MIEM, Some questions of cybernetic systems theory, issue 14
30. Galushkin AI (1971) Tailored recognition equipment. Works MIEM, Some questions of cybernetic systems theory, issue 14
31. Galushkin AI, et al. (1973) Requirements for planning and testing of equipment for automatic medical diagnostics. Digest of the Int. Conf. On Med. And Biol. Eng., Dresden
32. Galushkin AI (1974) Recognition signals on septrons. Publ. "Energy" monography
33. Galushkin AI (1974) Mathematic grounds of cybernetics. Publ. "High School"
34. Galushkin AI (1974) Multilayered pattern recognition systems and using possibility in medical diagnostics. Symposium "Theory and projection control system"
35. Galushkin AI (1976) Medical information systems on the computer. Thesis Russian conference "EVM-76" (in co-authorship)
36. Galushkin AI (1976) Requirements for planning and testing of equipment for automatic medical diagnostics. Medical radio-electronics, Coll. VNIIIMT, (in co-authorship)
37. Galushkin AI (1976) Signals recognition equipment. Cert. of Recognition No. 546910
38. Galushkin AI (1977) Diagnostics method of acquired heart diseases. Cert. of Recognition No. 562267
39. Galushkin AI (1979) Fluorography diagnostics equipment. Cert. of Recognition No. 686726 publ. 28.05. 1979
40. Galushkin AI (1980) Diagnostics method of acquired heart diseases. Cert. of Recognition No. 789103 publ. 21.08. 1980
41. Galushkin AI (n.y.) About a method of an adaptive diagnostics in threshold elements networks. Methods and tools of parallel diagnostics information. Lectures of All-Union school-seminar "Multisequencing of information processing", Lvov
42. Galushkin AI (1982) Some questions of progress fault-tolerance manager computing complexes in connection systems. Technical vehicles for communication, series "Vehicles for communication", issue 3
43. Galushkin AI (1990) Reconfiguration algorithms in multimicroprocessor systems. Cybernetic, AS USSR, No. 2, (in co-authorship)
44. Galushkin AI (1990) The architectures of neural computer. Joint British-Soviet Workshop on the transputer system Moscow, 26–29 June 1990
45. Galushkin AI (1990) The neural computers on the base of transputers and signal processors. Joint British-Soviet Workshop on the transputer system, Moscow, 26–29 June 1990
46. Galushkin AI (1990) The organization of studing and improvement of professional skills on the transputer system. Joint British-Soviet Workshop on the transputer system, Moscow, 26–29 June 1990 (in co-authorship)

47. Galushkin AI (1991) Neuromathematics: methods of object solutions on neurocomputers. Mathematical modeling 3, N8
48. Galushkin AI (1991) Neuron memory system (book 1, book 2). Publ. MAI, (in co-authorship)
49. Galushkin AI (1992) Neurocomputing technique. Auth. Ph. Wasserman, Publ. Mir
50. Galushkin AI (1992) Cytological image processing system on the transputer basis. Second Conference Transputer Systems and their application, (in co-authorship)
51. Galushkin AI (1992) Neuron EVM and neuromathematics (development conception). Transputer and neuron computers. Workshop Russian knowledge house
52. Galushkin AI (1992) Neuromathematics. Transputer and neuron computers. Workshop Russian knowledge house, (in co-authorship)
53. Galushkin AI (1992) Neuromathematics: problem-solving and algorithm procedure on neurocomputers. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-Don, Russia (in co-authorship)
54. Galushkin AI (1992) Adaptive neuronet algorithms for linear algebra decision problems. (p 1, p 2). RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-Don, Russia (in co-authorship)
55. Galushkin AI (1992) Sorting on neural networks. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-Don, Russia (in co-authorship)
56. Galushkin AI (1992) Neurocomputers and neuromathematics: information highway. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-Don, Russia
57. Galushkin AI (1992) The custom-made digital neurochip. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-Don, Russia (in co-authorship)
58. Galushkin AI (1992) Information base of publications on neurocomputers. RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers, Rostov-Don, Russia (in co-authorship)
59. Galushkin AI (1992) About joint Neurocomputers (introductory article for the editor-in-chief). Neurocomputer 1:3–7
60. Galushkin AI (1992) Methods of solving problems on neurocomputers. Neurocomputer 1:24–24, (in co-authorship)
61. Galushkin AI (1992) Adaptive neuronet algorithms for linear algebra decision problems. Neurocomputer 2:67–72, (in co-authorship)
62. Galushkin AI (1992) The custom-made digital neurochip. Neurocomputer N2:67–72 (in co-authorship)
63. Galushkin AI (1992) Structure optimization of multilayered neural networks with cross connections. Neurocomputer 3–4 (in co-authorship)
64. Galushkin AI (1993) Linguistics, neuronets and neurocomputers. Lecture on "Mathematical linguistics" seminar EVREKA, Nancy France, 15–17 February 1993
65. Galushkin AI (1993) Supertransputer EVM – bloc on desktop. "Poisk" 29.1–42.92 N5 (195)
66. Galushkin AI (1993) On the way to teraflops. Poisk 26.2–4.3.93 N9 (199)
67. Galushkin AI (1993) Transputer network-based general purpose neurocomputer. Int. Joint Conf. on Neural Networks Nagoja, Japan 25 October 1993 (in co-authorship)
68. Galushkin AI (1994) About problems of the sorting procedure using the neuronet. Neurocomputers 3–4:35–40
69. Galushkin AI (1994) About the edition of the collection history of neurocomputer development. Neurocomputers 3–4:66–76, (in co-authorship)
70. Galushkin AI (1995) The theory neuronet, neuromathematics and neurocomputers. Report on Conference Neurocomputers and their application. 15–18 February 1995
71. Galushkin AI (1995) Neural network algorithms of the decision of the special partial differential equation. Report on Conference Neurocomputers and their application. 15–18 February 1995
72. Galushkin AI (1995) Neural network algorithm decision graph theoretic problem. Report on Conference Neurocomputers and their application. 15–18 February 1995
73. Galushkin AI (1995) About the development of the SGI Neurochip-1 on the base of BMK Ispolin 60T. Report on Conf. Neurocomputers and their application. 15–18 February 1995, (in co-authorship)
74. Galushkin AI (1995) Foreign elemental base of neurocomputers and their using perspectives. Report on Conf. Neurocomputers and their application. 15–18 February 1995

75. Galushkin AI (1995) About the development of the complete set of neurochips for the computer complex Baget. Report on Conf. Neurocomputers and their application. 15–18 February 1995, (in co-authorship)
76. Galushkin AI (1995) Neurocomputers in the development of military technics in the USA. NCS RAS
77. Galushkin AI (1995) Results of the development of the neural network theory (1965–1995) in works of the neurocomputer science centre and prospects of its development. NCS RAS
78. Galushkin AI (1995) Neurocomputers in the development of military technics in the USA. Foreign radio electronics N5, N6
79. Galushkin AI (1996) Results of the development of the neural network theory (1965–1995) in works of the neurocomputer science centre and prospects of its development. Neurocomputer 1, 2
80. Galushkin AI (1996) Neural network theory, neuromathematics and neurocomputers. Report on the second Russian conference Neurocomputers and their application. 14–16 February 1996
81. Galushkin AI (1995) About the development of the superneurocomputer for the decision of problems of mathematical physics. Report on Conf. Neurocomputers and their application. 15–18 February
82. Galushkin AI (1996) About the development of the SGI Neurochip-1 on the base of BMK Ispolin 60T. Report on Conf. Neurocomputers and their application. 14–16 February 1996, (in co-authorship)
83. Galushkin AI (1996) About the common technique for the decision of problems in the neural network logical basis. Report on Conf. Neurocomputers and their application. 14–16 February 1996
84. Galushkin AI (1996) About manuals and monographies in the field of neurocomputers, prepared by employees of the Centre of neurocomputer science. Report on Conf. Neurocomputers and their application, 14–16 February 1996
85. Galushkin AI (1997) About the comparison of native and foreign techniques of adaptation in multilayered neural networks. Report on the third Conf. Neurocomputers and their application. 12–14 February 1997
86. Galushkin AI (1997) Actual problems of evolution in neurocomputer development. Report on the third Conf. Neurocomputers and their application. 12–14 February 1997
87. Galushkin AI (1997) Some conceptual questions of the neurocomputer's development. Foreign radio electronic 2, (in co-authorship)
88. Galushkin AI (1997) About modern directions of the neurocomputer's development. Information technology 5
89. Galushkin AI (1997) About modern directions of the neurocomputer's development. Information technology 1,2 (5–22)
90. Galushkin AI (1997) Neuronet algorithms of an optimum choice of a subset of vectors of a causal multivariate sample. Neurocomputer 1, 2
91. Galushkin AI (1997) Results of the third Russian Conference Neurocomputers and their application. Neurocomputer 3, 4
92. Galushkin AI (1998) Results of International conference MICRONEURO'97. Report on the fourth Russian conference "Neurocomputers and their application" 18–20 February 1998
93. Galushkin AI (1998) The concept of the development of the SGI-neurochip on the basis of native technology. Report on the fourth Russian conference Neurocomputers and their application, 18–20 February 1998 (in co-authorship)
94. Galushkin AI (1998) About the development of the program "Neuromathemathic". Report on the fourth Russian conference Neurocomputers and their application, 18–20 February 1998
95. Galushkin AI (1998) Modern line of the development of neurocomputers in Russia. Foreign radio electronic 1
96. Galushkin AI (1998) Performance evaluation of neurocomputers. Foreign radio electronic 1, (in co-authorship)
97. Galushkin AI (1999) The neurocomputer is a promising supercomputer. Workshop of "Neuromathematic-99", Debate about neurocomputers (84–89)
98. Galushkin AI (1999) Neural network theory and neuromathematics. IJCNN'99, Washington, July 10–16 1999
99. Galushkin AI (1999) Debate about neurocomputers – 10 years later. Neurocomputer 1

100. Galushkin AI (1999) Neurocontrol: main principles and directions of application neurocomputers for the decision of problems of dynamic object management. Neurocomputers 1, (in co-authorship)
101. Galushkin AI (1999) Neurocomputers of 80th (the beginning of the next revolution in neurocomputers). Foreign radio electronic 1
102. Galushkin AI (1999) Prospects of application neurocomputer technologies in space systems. Report on the fifth Russian conference of Neurocomputers and their application, 17–19 February 1999 (in co-authorship)
103. Galushkin AI (1999) Neuro methods of space image processing. Report on the fifth Russian conference Neurocomputers and their application, 17–19 February 1999 (in co-authorship)
104. Galushkin AI (1999) Prospects of application neurocomputer technologies in professional medicine. Report on the fifth Russian conference Neurocomputers and their application, 17–19 February 1999 (in co-authorship)
105. Galushkin AI (1999) "Neuromathematic" – program package for mathematical problem decision on a neural basis. Report on the fifth Russian conference Neurocomputers and their application, 17–19 February 1999 (in co-authorship)
106. Galushkin AI (1999) Digital neurochip. Foreign radio electronic 1, (in co-authorship)
107. Galushkin AI (2000) Neural network theory. Collection Neurocomputers and their application, book 1. Russia, Moscow, Radiotechnika
108. Galushkin AI (2000) Neurocontrol and their application (translation from English). Collection Neurocomputers and their application, book 2. Russia, Moscow, Radiotechnika, (in co-authorship)
109. Galushkin AI (2000) Neurocomputers. Collection Neurocomputers and their application, book 3. Russia, Moscow, Radiotechnika
110. Galushkin AI (2000) Prospective problems of neuronet theory. Neurocomputer 3
111. Galushkin AI (2000) Neurocontrol. Report on the sixth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
112. Galushkin AI (2000) Prospects of neurocomputer information technologies in space means. Report on the sixth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
113. Galushkin AI (2000) Neurocomputers in control of helicopters. Report on the fifth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
114. Galushkin AI (2000) About ICNN'99 (Washington 1999). Report on VI Russian conference Neurocomputers and their application, 16–18 February 2000
115. Galushkin AI (2000) Neurocomputers in CALL-centre. Report on VI Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
116. Galushkin AI (2000) The decision of a problem of extrapolation non-negative function on a neural network with the fixed structure with full consecutive connections. Report on VI Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
117. Galushkin AI (2000) The account of artifacts at the formation of the training sample for a problem of function extrapolation. Report on the sixth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
118. Galushkin AI (2000) Substantiation of necessity of feedback introduction and the decision of a problem of non-negative function extrapolation on a three-layer neural network of the fixed structure with feedback. Report on the sixth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
119. Galushkin AI (2000) Some historical aspects of the development of the elemental base of computing systems with mass parallelism (1980s–1990s). Report on the sixth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
120. Galushkin AI (2000) On the debate about neurocomputers. 10 years later. Report on the sixth Russian conference Neurocomputers and their application, 16–18 February 2000 (in co-authorship)
121. Galushkin AI (2000) Some historical aspects of development of the elemental base of computing systems with mass parallelism (1980s–1990s). Neurocomputers 1
122. Galushkin AI (2000) Modern directions of the development of neurocomputer technologies in Russia. The Computer chronicle 3

123. Galushkin AI (2000) Neural network algorithms for the optimal selection of the vector subset from the random multidimensional sample. Neurocomputers: design and applications 1, 2:1–15
124. Galushkin AI (2000) Neural network algorithms for function extrapolation and their Application in CALL-center Forecasting Tasks, Part 1. Neurocomputers: design and applications 1, 3:9–33, (in co-authorship)
125. Galushkin AI (2000) Neural network algorithms for function extrapolation and their Application in CALL-center Forecasting Tasks, Part 2. Neurocomputers: design and applications 1, 4, (in co-authorship)
126. Galushkin AI (2000) Neural network algorithms for function extrapolation and their Application in CALL-center Forecasting Tasks, Part 3. Neurocomputers: design and applications 1, 5:12–26, (in co-authorship)
127. Galushkin AI (2000) Problems of neural network theory in their perspective. Neurocomputers: design and applications 1 6:2–15
128. Galushkin AI (2000) Supercomputers and neurocomputers. Neurocomputers: design and applications 1, 6:22–34
129. Galushkin AI (2001) Sphere of the neurocomputer's application. Works of the International confederation Intellectual and multiprocessing systems IMS, 1–6 October 2001
130. Galushkin AI (2001) Fizteh and the Russian agency on control systems – on the development of neurocomputers. Magazine the artificial intelligence 3:391–410
131. Galushkin AI (2001) Neural networks: training, the organization and application. Series Neurocomputers and their application the book 5. Publishing house Radio engineering, (in co-authorship)
132. Galushkin AI (2001) Neural networks: history of the development of the theory. Series Neurocomputers and their application the Book 5. Radio engineering, (in co-authorship)
133. Galushkin AI (2001) Neural network algorithms for function extrapolation and their application in CALL-center forecasting tasks, part 3. Neurocmputer 1:10–21 (in co-authorship)
134. Galushkin AI (2001) Supercomputers and neurocomputers. Neurocomputer 6:16–25
135. Galushkin AI (2001) The sphere of application neurocomputers extends. Neurocomputer 1:60–70
136. Galushkin AI (2001) Methods for realization of invariance to linear transformations at recognition of bidimentional images. The appendix to magazine Information technologies 1, (in co-authorship)
137. Galushkin AI (2001) The sphere of application neurocomputers extends. The appendix to magazine Information technologies 10
138. Galushkin AI (2001) Fizteh the Russian agency on control systems – on the development of neurocomputers. Fizteh: a sight in the future of p 425–458. M, AST
139. Galushkin AI (2001) Problem of neural network theory in their perspective. Int. Conf. On Neural Information Processing Shanghai, ICONIP'01
140. Galushkin AI (2001) Supercomputers and neurocomputers. Int. Conf. On Neural Information Processing Shanghai, ICONIP'01
141. Galushkin AI (2001) Physical model of spatial system active vibration isolation and promptings. The patent for the invention 2224295 (in co-authorship)
142. Galushkin AI (2001) "Neurocomputer" and "Neurocomputers: design and applications" on a boundary of a new millennium. Report on the eighth Russian conference Neurocomputers and their application, 14–16 February 2001, (in co-authorship)
143. Galushkin AI (2001) Supercomputers and neurocomputers. Report on the eighth Russian conference Neurocomputers and their application, 14–16 February 2001
144. Galushkin AI (2001) System active vibration isolator and high-frequency prompting of large-sized aerials of space radio telescopes with neurocontrol. Report on the eighth Russian conference Neurocomputers and their application, 14–16 February 2001 (in co-authorship)
145. Galushkin AI (2001) Problems of neural network theory in their perspective. Report on VIII Russian conference Neurocomputers and their application, 14–16 February 2001
146. Galushkin AI (2001) The field of neurocomputer application expands. Neurocomputer design and applications 2,1:2–18
147. Galushkin AI (2002) Neurocomputer processing of the images in the task of tuberculosis contagion identification. Int. Conf. On Neural Information Processing Singapore, ICONIP'02 (in co-authorship)

148. Galushkin AI (2002) Perspective of neurocomputers in biometric identification systems. Workshop of first biometric conference in Russia, Moscow, 10 December 2002, (in co-authorship)
149. Galushkin AI (2002) The Complex neuronet algorithms and programs for processing of the images received from a fluorescent microscope, with the purpose of microscopic object recognition (neurodiagnostic). The Certificate on official registration of the computer program 2002611587, (in co-authorship)
150. Galushkin AI (2002) Problem of Neural Networks Theory in Their perspective. Workshop International Conference Mathematical Methods in intellectual information systems, Smolensk on 16–17 May 2002
151. Galushkin AI (2002) Biomolecular neural device. Neurocomputers and their application, book 33. Radiotechnika, (in co-authorship)
152. Galushkin AI (2002) Neuromathematic. Neurocomputers and their application, book 6. Radiotechnika, (in co-authorship)
153. Galushkin AI (2002) Fundamentals of neurocontrol. The appendix to magazine Information technologies 10
154. Galushkin AI (2002) Neural network control systems. Neurocomputers and their application, book 8. Radiotechnika, (in co-authorship)
155. Galushkin AI (2002) Fundamentals of neurocontrol. Neurocomputer 9, 10:87–106
156. Galushkin AI (2001) About prospects in the development and application of neurocomputers. Report on the eighth Russian Conference Neurocomputers and their application, 21–22 March 2001
157. Galushkin AI (2002) About the book "Application of neurocomputers". Report on the eighth Russian conference Neurocomputers and their application, 21–22 March 2002
158. Galushkin AI (2002) Development and research of neural network methods and algorithms for space image processing. Report on the eighth Russian conference Neurocomputers and their application, 21–22 March 2002, (in co-authorship)
159. Galushkin AI (2002) Vibration isolator and precision prompting gravity and inertial sensitive space designs. Report on the eighth Russian conference Neurocomputers and their application, 21–22 March 2002, (in co-authorship)
160. Galushkin AI (2002) The way and means of realization neural network algorithms. Report on the eighth Russian conference Neurocomputers and their application, 21–22 March 2002, (in co-authorship)
161. Galushkin AI (2001) Neural network algorithms for the decision of the linear algebraic equation systems. Report on the eighth Russian conference Neurocomputers and their application, 21–22 March 2001
162. Galushkin AI (2003) Neural network theory. Publishing house of university Sinhua Pekin, (in Chinese)
163. Galushkin AI (2003) Neurocomputers in processing image systems. Neurocomputers and their application, book 7, Radiotechnika, (in co-authorship)
164. Galushkin AI (2003) Neurocomputers in residual classes. Neurocomputers and their application, book 7, Radiotechnika, (in co-authorship)
165. Galushkin AI (2003) Neurocomputer for processing signal systems. Neurocomputers and their application, book 9, Radiotechnika, (in co-authorship)
166. Galushkin AI (2003) Neurocomputers for the decision of regional problems of the field theory. Neurocomputers and their application, book 10, Radiotechnika, (in co-authorship)
167. Galushkin AI (2003) Intellectual systems. Neurocomputers and their application, book 12, Radiotechnika, (in co-authorship)
168. Galushkin AI (2003) Neurocontrol construction and systems. Neurocomputers and their application, book 13, Radiotechnika, (in co-authorship)
169. Galushkin AI (2003) Neuromathematic (problems of development). Neurocomputer 1
170. Galushkin AI (2003) Paralleling methods and hardware-software realization neural network algorithms for image processing. Neurocomputer 1:3–21, (in co-authorship)
171. Galushkin AI (2003) Neurocomputer processing of the images in the task of tuberculosis contagions identification. Neurocomputer 1:45–52, (in co-authorship)
172. Galushkin AI (2003) Application of neural networks in chemistry and chemical technology. Neurocomputer 1:66–107, (in co-authorship)

173. Galushkin AI (2003) Perspective of neurocomputers in biometric identification systems. Neurocomputer 5:39–61, (in co-authorship)
174. Galushkin AI (2004) Neurocomputers and their application in China on a boundary of a millennium (b. 1 and b. 2). Publishing house the Hot line – a TV set of M
175. Galushkin AI (2003) Neurocomputers in aircraft (planes). Series Neurocomputers and their application the book 14, publishing house the Radio engineering of M, (in co-authorship)
176. Galushkin AI (2003) Neural network algorithms for biometric identification. Neurocomputers and their application book 15, publishing house the Radio engineering of M, (in co-authorship)
177. Galushkin AI (2003) Neurocomputers with parallel architecture. Neurocomputers and their application book 16, publishing house the Radio engineering of M, (in co-authorship)
178. Galushkin AI (2003) Neurocomputers in space technics. Neurocomputers and their application book 17, publishing house the Radio engineering of M, (in co-authorship)
179. Galushkin AI (2001) Neurocontrol. Base direction for the development of the theory and practice of management by complex dynamic systems (the plenary report). Workshop the second International conference Parallel processing and management problem, (in co-authorship)
180. Galushkin AI (2004) Neural networks in problems of recognition of images and processing of images (the plenary report). Workshop the seventh International conference Recognition of images and processing of images: new information technologies 2004, St.-Petersburg on 18–23 October 2004
181. Galushkin AI (2004) Neurocomputers in problems of image recognition, processing of signals, images and management of dynamic systems (the plenary report). The International conference Supercomputer systems and their application Minsk SSA' on 26–28 October 2004
182. Galushkin AI (2004) System and a method of adaptive neural network computing of granulometric composition for crushed, balled and\or granulated material. The Application for patenting No. 2004128312, 23.9. 2004 (in co-authorship)
183. Galushkin AI (2005) Neurocomputers for the decision problems of information safety maintenance. The Report at the St.-Petersburg conference Information safety of regions of Russia
184. Galushkin AI (2005) Formation of entry conditions for acceleration of adjustment of weight factors of neural networks. Neurocomputer 5
185. Galushkin AI (2005) Artificial intelligence problem and the neurocomputer. Neurocomputer 6
186. Galushkin AI (2005) Transputer systems – the beginning of the development in Russia of the computer with mass parallelism. Neurocomputer, No. 3, (in co-authorship)
187. Galushkin AI (2005) Certification and standardization of biometric identification systems. Materials of the International scientific and technical conference Intellectual and multiprocessing systems, Divnomorsk, Russia, (in co-authorship)
188. Galushkin AI (2005) Neurocomputers for the decision of problems of maintenance of information safety. Materials of the International scientific and technical conference Intellectual and multiprocessing systems, Divnomorsk, Russia
189. Galushkin AI (2005) About modern problems of neurocontrol for complex dynamic systems. First Workshop on Russian Section of IEEE Computational Intelligence Society
190. Galushkin AI (2005) Neurocontrol in China on the boundary of a millennium. First Workshop on Russian Section of IEEE Computational Intelligence Society
191. Galushkin AI (2005) Neurocomputers for the decision problems of information safety maintenance. Neurocomputer 12
192. Galushkin AI (2005) About ways of development of integrated neuronet systems for robot control. "International program on perspective robotics" Conference "Adaptive intelligence robots: modern state and perspective", Moscow, 24–26 November 2005
193. Galushkin AI (2005) About the technique for the decision of problems on a neural network logical basis. Second workshop of Russian section of IEEE computational Intelligence Society, Moscow, 16.11. 2005
194. Galushkin AI (2005) Problem solving for function approximation in a small sample. Second workshop of Russian section of IEEE computational Intelligence Society, Moscow, 16.11. 2005
195. Galushkin AI (2005) The development of the technique for the decision of problems on a neural network logical basis. International Society conf "Neurotechnologies and their applications", Kramatorsk

196. Galushkin AI (2005) Information safety on neurocomputers systems. International Society conf. "Neurotechnologies and their applications", Kramatorsk
197. Galushkin AI (2005) Principles of formation of entry conditions for acceleration of adjustment of weight factors of neural networks. International Society conf "Neurotechnologies and their applications", Kramatorsk
198. Galushkin AI (2005) Construction of high-accuracy measuring device based on neurocomputers: the developmental principles. International Society conf "Neurotechnologies and their applications", Kramatorsk
199. Galushkin AI (2005) Construction of high-accuracy measuring device based on neurocomputers: the developmental principles. Third Workshop "Biometric System" of Russian Section of IEEE CIS, Moscow, 14.12. 2005
200. Galushkin AI (2005) Neurocomputers in biometric identification systems. Third Workshop "Biometric system" of Russian section of IEEE Computational intelligence society, Moscow, 14.12. 2005
201. Galushkin AI (n.y.) Research of application technologies for engineering problems with tight systems of equations and giant-scale number unknown. Draft Project Goscontract FAN code IT–13.3/001
202. Galushkin AI (2006) About methods of solving problems on a neural net logical basis. Reports the eighth Russian Scientific-technical conference "Neuroinformatics', 24–27 January 2006, MIFI, Moscow
203. Galushkin AI (2006) Formation of beginning terms for acceleration of neural net coefficient tooling in optimization problems. Reports the eighth Russian Scientific-technical conference "Neuroinformatics", 24–27 January 2006, MIFI, Moscow
204. Galushkin AI (2006) Development of an integrated neural net system of perspective robot control. Russian scientific-practical conference "Perspective systems and control problems" Dombai, 13–17 March 2006
205. Galushkin AI (2006) Development of an integrated neural net system of perspective robot control. Russian seminar "Actual methods of navigation and motion control", Institute of control problems RAN, 18–19 April 2006
206. Galushkin AI (2006) Integration of controlling functions of actual robotechnical systems. Conference MIREA, 15–24 May 2006
207. Galushkin AI (2006) Neural net technologies – base of the integrated control systems of perspective robot construction. MIREA, Scientific-practical seminar "Perspective problems and control systems", June 2006
208. Galushkin AI (2006) Neurocomputers: from the program to apparatus realization. Int. conference "Development of computing technique in Russia and ex GUS countries: history and perspectives" SORUCOM-2006, Petrozavodsk, 3–7 June 2006
209. Galushkin AI (2006) Transputer systems – the beginning of making EVM with mass parallelism in Russia. International Conference "Development of computing technique in Russia and ex GUS countries: history and perspectives" SORUCOM-2006, Petrozavodsk, 3–7 June 2006
210. Galushkin AI (2006) Neural network recognition of spherical body set grain-size distribution using envelope of surface. IEEE World Congress on Computational Intelligence Vancouver, B.C. Canada, July 16–21 2006
211. Galushkin AI (2006) Neural net solution algorithms of complex mathematic problems and cluster EVM based on graphic processors. The third International Conference "Parallel Computing and control problems" 2–4 October 2006. Institute of control problems named academician V. A. Trapeznikov
212. Galushkin AI (2006) Neurocomputers: from the program to apparatus realization. Monography, Publ. "Hot-line-telecom"
213. Galushkin AI (2006) Neural network theory second edition, remake, add. Monography, Publ. "Radio-technique", Moscow
214. Galushkin AI (2006) History of neural network development second edition remake, add. Collection of articles, Publ. "Radio-technics", Moscow
215. Galushkin AI (2006) Methods of solving problems on a neural net logical basis. Attachment to Intellectual Technologies 6
216. Galushkin AI (2006) Problems in neuronet logical basis solution. Neurocomputer 2:49–70

Index

A

activation 39, 40, 358
  –, slope 363
adaptation
  –, algorithm 364
  –, neural network 27
adjustment
  –, algorithm 195, 224, 257, 293
    –, analytical 226
  –, closed cycle 170, 273
  –, neural network 207
    –, continuum 189
  –, dynamics 237, 238, 240–242
  –, neuron 190
  –, process 243
  –, recurrent 201, 203, 205
algorithm
  –, adaptation 25, 155, 364
  –, adjustment 195, 224, 257, 293
    –, analytical 226
  –, convergence 280
  –, deterministic selection 208
  –, failure localization 323
  –, learning 198, 277, 283, 290
    –, sequential 273
  –, neural 2, 3
    –, implementation 7
  –, problem
    –, solution 343
    –, solving 17
  –, search 143, 147, 224
    –, development 155
    –, extremum 153
    –, multivariable functional extremum 25
    –, random 154
  –, self-learning 294
  –, test design 331
analysis
  –, matrix 148
  –, stage of 264
analytical system 360
approach
  –, algorithmic 22
  –, neurophysiological 21
  –, psychological 21
  –, systematic 22
approximation
  –, constant 197
  –, linear 197
  –, method, stochastic 146
architecture of neural network 43
automata theory 4

B

backward connection 239

C

characteristics, topological 60
class
  –, continuum 354
  –, number 113
closed cycle adjustment 170, 273
closed-loop
  –, neural network 180, 223
clusterization 220
CMOS-neurochip 3
coefficient
  –, adjustment 247
  –, readjustment 12
  –, setting 216
complexity criterion 53
computer
  –, class 5
  –, CRAY type 5
  –, development 1
    –, economic factors 8
  –, Elbrus 6
  –, engineering 5

–, fine-grained 6
–, large-grained 6
–, MIMD architecture 2
–, neural 1, 6, 19
–, capacity 13
–, classification 12
–, development 1
–, elemental base 14, 32
–, implementation 3, 32
–, modularity 9
–, position 5
–, preparation 21
–, type 12, 13
–, personal 10
–, science, Russian 16
–, SIMD architecture 2, 3
–, SISD architecture 3
–, universalism 8, 9
–, von-Neumann 2
computing facility class 5
condition
–, initial 207, 210, 211, 213, 216, 248, 363
–, selection method 207
connection
–, backward 44, 49, 50
–, cross 49, 54
–, direct 43
–, lateral 45, 357
–, random 357
–, sequential 45, 47
Connection Machine 3
constant pproximation 197
continual neural network 67
continuum
–, model 135
–, neural layer 205
–, neuron 68, 198
–, classification 69
–, layer 190, 195, 199, 200, 202
–, layer model 69
–, of classes 354
–, of features 71
–, of solutions 101, 102
–, pattern class 86
–, solution 137, 139
control theory 5
CRAY type computer 5
CRAY XMP 3
criterion
–, minimum 133, 134
–, optimization 132
cross connection 54
–, optimization 53
CYBER 205 3

D

derivative
–, estimation 185
–, second 363
design of neural network 170
designation unification 367
deterministic
–, selection 208, 213
development
–, closed-cycle adjustable neural network 182
–, neural network closed-loop 180
–, search algorithm 155
diagnostics 29, 303, 321
distribution
–, error 121
–, law 83, 84, 121
–, multi-modal 132
divisional surface 90

E

element, boolean 35
engineering method 257
equation
–, differential 2
–, Euler 2
–, Navier-Stokes 2
–, Poisson 2
error
–, distribution 121
–, function 99, 103
–, probability 280
estimation
–, matrix 202
–, quality improvement 277
Euler equation 2
extrapolation 362
extrema
–, global 154
–, local 154
–, search 143, 146
–, algorithm 143, 153, 361
–, iteration 148, 152
–, parameters 361

F

failure
–, catastrophic 317
–, diagnostics, method 332
–, localization algorithm 323
feature
–, continuum 71–73

–, informative 28
–, selection 295, 297
–, informative 296, 297, 299, 300, 302
–, problem 295
–, space 69, 78, 91
feedback 240–242
filtering 362
five-feature space 198
Fourier transformation 2
function
–, activation 39, 358
–, slope 363
–, approximation 218
–, error 99, 103
–, extrapolation 218
–, extrema 277
–, search 143, 146
–, Lagrange 148
–, loss 107
–, multivariable 154
–, risk 132, 177, 178, 277
–, minimum average 175, 177, 178
–, weighting 195, 197, 202, 203, 205
functional
–, optimization 132, 143, 225, 360
–, generation 360
–, secondary 185, 198, 363

H

hardware
–, emulation 3
–, implementation 359
hyperplane 50, 51, 310, 312, 313, 324, 325, 333
–, number 280, 281
–, position 253–255, 257

I

initial condition 207, 210, 211, 213, 216, 248, 363
–, selection 207, 212
input signal 113, 207, 219, 221, 224
–, characteristics 77
–, generation 352
–, nonstationary 87, 354
–, pattern 79
–, probability 84
–, distribution 79
iteration
–, extremum search 152
–, method 146
–, process stability 150

K

K^* value 226
K_p solution 64, 138

L

Lagrange function 148
layer, open-loop 205
learning
–, algorithm 198, 277, 283, 290
–, sequential 273
–, method 292
–, mode 183, 219
–, harm 354
–, procedure 190, 193, 195
linear
–, approximation 197
–, equality system 217
–, inequality system 217
logic
–, continuous 38
–, mathematical 4
–, multi-threshold 37
–, threshold 4
logical tree 276
loss
–, coefficient 95
–, function 107

M

mathematical
–, logic 4
–, statistics 4
mathematics
–, computational 5, 341
–, neural 17
matrix
–, analysis 148
–, estimation 202
–, inversion 258
–, problem 258
–, parameter 257
–, transformation 282
method
–, analytical 263
–, engineering 257
–, failure diagnostics 332
–, functional minimization 285, 288
–, iteration 146
–, extremum search 151, 152
–, learning 292
–, problem solving 341

–, reliability investigation 305
–, selection 207
–, stochastic approximation 146
–, synthesis 292
MIMD (Multiple Instruction – Multiple Data) 2, 3, 6–8, 10, 11, 17, 29, 36
minimization, neuron number 300
minimum criterion 133, 134
mode
–, learning 219
–, self-learning 113, 140, 217
model
–, continuum 135
–, definition 89
–, mathematical 79
–, optimal 75, 104, 110
–, structure 89
–, self-learning mode 116
modelling, dynamic system 220
modularity 9
MSIMD (Multiple variant of SIMD) 6, 14
multi-threshold 37
myths, neural network theory 368

N

Navier-Stokes equation 2
network, neural 21, 77, 250
–, adaptation 27
–, algorithm 155
–, adaptive 161
–, diagnostic 337
–, adjustment 195, 207
–, algorithm 163
–, closed-cycle 182, 263
–, algorithm 18, 341
–, analysis 223
–, architecture 43
–, backward connections 239
–, cell-like 357
–, classification 90
–, closed-cycle adjustment 182, 263
–, closed-loop 180, 183
–, multilayer 223
–, coefficient setting 216
–, complex 358
–, connections
–, amorphous backward 44
–, cross 44, 53
–, direct 43
–, lateral 45, 357
–, ordered backward 44
–, random 357
–, sequential 47
–, continual 67, 358
–, continuum 189, 205
–, model 135
–, description 47
–, design 170, 258
–, diagnostics 29, 303, 321
–, dynamics 231
–, errors 121
–, feedback loop 357
–, functional reliability 308
–, functioning 322, 323
–, informative feature selection 295
–, initial conditions 212
–, input signal 24, 77, 221
–, interval 358
–, investigation 262
–, approaches 21
–, K_p solution 138
–, layer 68
–, learning mode 235, 242
–, logical basis 344, 349
–, main types 45
–, model 104, 110, 116
–, multilayer 22, 28, 31, 45, 51, 169, 178
–, open-loop 25, 121
–, optimal model 89
–, optimization 24, 64, 114
–, output channel 139
–, parametrical reliability 309
–, reliability 29, 303, 305, 317
–, self-learning mode 140, 256, 302
–, solution continuum 135, 137
–, state graph 322
–, structure 23, 28, 33, 356, 359
–, fixed 352
–, flexible 350
–, variable 358
–, synthesis 223, 273
–, successive 292
–, theory 343, 368
–, two-layer 54, 167
–, continuum 198
–, type 89, 90, 96, 98, 105–109
–, verification 364
Neumann architecture 6
neural
–, computer
–, adequate problems 10
–, definition 5
–, network, model 75
neuromathematics 341
neuron 35, 190, 193

–, adjustment 190
–, algorithm 341
–, learning 290
–, continuum 68, 69, 198–200
–, adjustment 190
–, classification 69
–, input feature 67
–, solution 38, 164, 169, 175, 268
–, diagram 36
–, discrete set 69
–, failures 331
–, investigation 224
–, K_p solution 126, 175, 248
–, layer 71, 202, 246
–, coefficient 212
–, continuum 196
–, experiments 236, 237
–, feature continuum 71, 73
–, multi-channel 72
–, output values 72
–, structure 202
–, weighting function 197
–, number
–, minimization 300
–, output 331
–, solution 121, 124, 173
–, continuum 164, 169, 175, 268
–, synthesis 285, 290
–, tolerance 37
node 21
–, capacity 7
number
–, regions 55
–, transformation 261

O

open-loop
–, layer 205
–, structure 200
–, network 121
–, structure 195, 200, 202, 203
operation speed 149
–, optimal 149
–, optimization 148
operator 276, 277
optimal model 104
–, definition 89
optimality conditions 152
optimization
–, criterion 97, 132
–, cross connection 53
–, functional 129, 131, 132, 225
–, secondary 143, 363
–, primary 114, 115, 173, 355
–, problem 55, 57, 216
–, secondary 129, 155
–, speed 148
–, structural 60
output
–, channel 139, 245
–, signal 196
–, generation 355

P

parallelism
–, algorithmic 3
–, events 3
–, geometric 3
parameter matrix 190, 193
pattern recognition 128, 218
–, non-stationary 231
plan experiment 365
Poisson equation 2
possibility, qualitative 9
primary optimization 114, 115, 173
probability 354
problem 163
–, algorithm 17
–, class 10
–, complexity criterion 53
–, formalized 10
–, matrix inversion 258
–, non-formalized 10
–, optimization 55
–, selection 344
–, solution
–, algorithm 343
–, program package 349
–, solving 17, 341, 349
–, methods 341
–, statement 77
–, traveling salesman 220
–, weakly formalized 10
procedure, learning 195
process iteration 150
processor
–, signal 3
–, Weitek 3

Q

qualification
–, supervisor 353
–, teacher, arbitrary 116, 140

R

random search 154, 277, 279
  –, algorithm 154
readjustment of coefficient 12
recognition
  –, pattern 128
  –, probability 313–316
recurrent adjustment 203, 205
region number 55
reliability 29, 303, 305
  –, functional 306
  –, investigation 305
  –, parametrical 309
risk function 132, 175, 177, 178, 277
  –, minimum criterion 132

S

search
  –, algorithm 143, 147, 224
    –, development 155
  –, distribution mode center 242
  –, extremum 148
    –, algorithm 153
    –, function 143, 146
  –, random 154, 277, 279
  –, system 360
selection
  –, deterministic 208, 211, 213
  –, method 207
self-learning mode 115, 140, 217
signal
  –, generation 352, 355
  –, input 77, 113, 207, 219, 221, 224
  –, multidimensional 110
  –, output 196
  –, self-learning mode 113
  –, types 70, 71
SIMD (Single Instruction – Multiple Data) 5, 6, 11, 13–15, 23
software emulation 3
solution 121
  –, continuum 124, 135, 137, 139
  –, discrete 101
  –, K_p 64
  –, process 21
  –, task 356
space, initial 299
speed
  –, operation 149
  –, optimization 148
state graph 322–325, 331, 333–335
statistics, mathematical 4
structural optimization 60
structure
  –, neural network 23
  –, open-loop 195, 200, 202, 203
    –, layer 200
supercomputer 3
  –, fine-grained transputer-type 8
supervisor qualification 353
surface
  –, divisional 90, 208, 236, 277
    –, linear 46
    –, nonlinear 128
    –, piecewise linear 274, 275
synthesis methods 292
system
  –, adjustment 226, 227
    –, closed-cycle 233–235
  –, analytical 360
  –, approach 22
  –, binary 261
  –, decimal 261
  –, linear
    –, equality 217
    –, inequality 217
  –, modelling, dynamic 220
  –, search 360

T

teacher qualification
  –, arbitrary 140, 183, 262
  –, subjective 262
  –, unequal 82
theory control 5
threshold 35
  –, function
    –, tables 290
  –, linear element 35
  –, logic 4
  –, multi- 37
  –, value 37
transputer 8
  –, network 3
traveling salesman problem 220

V

verification 364

W

weighting
  –, coefficient 37, 50
  –, function 195, 197, 202, 203, 205